aws

Panduan Pengguna

Amazon Neptune

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Neptune: Panduan Pengguna

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu Neptune? 1
Pembaruan Teranyar 4
Memulai dengan Neptunus 70
Apa itu basis data grafik? 70
Mengapa menggunakan grafik? 71
Aplikasi basis data grafik 72
Bahasa kueri grafik 76
Contoh kueri 76
Kursus Neptune Online 78
Menggali lebih dalam 78
Menggunakan Neptunus dengan notebook grafik 79
Menggunakan Workbench Neptune 80
Menyebarkan dengan CloudFormation 85
Mengaktifkan log CloudWatch 85
Hosting lokal 87
Migrasi ke 4 JupyterLab 88
Prasyarat 92
Konfigurasi siklus hidup 93
Sinkronisasi dari snapshot 93
Setelah pembuatan notebook 93
Buat buku catatan 94
Menyiapkan notebook Neptunus secara manual 95
Neptunus grafik buku catatan sihir 96
Injeksi variabel 98
Argumen kueri umum 98
%seed 100
%load 100
%load_ids 100
%load_status 101
%cancel_load 101
%status 101
%gremlin_status 101
%opencypher_status, atau %oc_status 101
`%stream_viewer` 102

%sparql_status 103
%graph_notebook_config 103
%graph_notebook_host 103
%graph_notebook_version 103
%graph_notebook_service 103
%graph_notebook_vis_options 104
%statistik 104
%ringkasan 105
%reset_graph 105
%get_graph 105
%DegreeDistribusi 105
%%sparql 106
%%gremlin 107
%%opencypher, atau %%oc 108
%%graph_notebook_config 109
%%graph_notebook_vis_options 109
%neptune_ml 110
%%neptune_ml 114
Visualisasi data grafik 116
Antarmuka grafik 116
Visualisasi Gremlin 118
Visualisasi SPARQL 119
Tutorial visualisasi 120
Memecahkan masalah notebook 121
Desinkronisasi status widget 121
JupyterLab 4.x penundaan layar pemuatan awal 121
Masalah akses Graph Explorer 122
Kiat umum 122
Pengaturan Neptune 123
Jenis instans DB 124
Alokasi sumber daya instans 124
t3 dan t4g 125
Instans r4 126
Instans r5 126
Instans r5d 126
Instans r6g 127

Instans r6i 127
Instans x2g 127
Instans x2iezn 128
Instans x2iedn 128
Instans r8g 128
Instans r7g 129
Instans r7i 129
Instans serverless 130
Jenis penyimpanan 131
I/O — Penyimpanan yang dioptimalkan 131
Buat cluster Neptunus 132
Prasyarat 134
Buat klaster 139
Konfigurasikan VPC 142
Tambahkan subnet 143
Konfigurasikan VPC 143
Membuat grup keamanan 144
DNS di VPC Anda 145
Connect ke sebuah cluster 145
Mengatur curl atau awscurl 146
Cara untuk terhubung 146
Di VPC 147
Dari VPC yang berbeda 149
Melalui jaringan pribadi 150
Titik Akhir Publik 150
Mengamankan akses ke Neptunus 160
Kebijakan IAM 161
Grup keamanan VPC 161
Autentikasi IAM 161
Mengakses data grafik 163
Menyiapkan curl 146
Bahasa kueri 163
Menggunakan Gremlin 164
Menggunakan OpenCypher 169
Menggunakan SPARQL 170
Memuat data 171

Memantau cluster Neptunus 171
Memecahkan masalah cluster Neptunus 172
Neptunus Tanpa Server 173
Kasus penggunaan tanpa server 173
Batasan 174
Penskalaan kapasitas 175
Mengatur minimum 177
Mengatur maksimum 177
Perkirakan pengaturan kapasitas 178
Konfigurasi tambahan 180
Konfigurasi campuran 180
Menetapkan tingkatan promosi 180
Penyelarasan pembaca-penulis 181
Hindari nilai batas waktu yang sangat besar 182
Mengoptimalkan konfigurasi Anda 182
Menggunakan Tanpa Server 183
Membuat cluster tanpa server 183
Konversi ke Tanpa Server 184
Memodifikasi rentang kapasitas 185
Mengubah instance menjadi provisioned 185
Konfigurasi klien 185
Memantau 186
Database global Neptunus 188
Ikhtisar 188
Keuntungan 189
Batasan 190
Menyiapkan database global 191
Persyaratan konfigurasi 191
Membuat global database 192
Gunakan cluster DB yang ada sebagai primer 194
Menambahkan wilayah sekunder 194
Terhubung 196
Mengelola database global 196
Menghapus cluster 197
Menghapus global database 197
Memodifikasi database global 198

Pemulihan bencana di database Neptunus 198
Lepaskan dan promosikan 199
Kegagalan terencana yang dikelola 201
Memantau penyebaran Neptunus 202
Ikhtisar Neptunus 204
Kepatuhan Standar 205
Kepatuhan standar Gremlin 205
Kepatuhan standar SPARQL 221
Kepatuhan spesifikasi OpenCypher 228
Model Data Grafik 242
Kamus 242
Strategi Pengindeksan 243
Cache pencarian 247
Gunakan kasus untuk cache pencarian 247
Menggunakan cache 248
Semantik Transaksi 250
Tingkat Isolasi 250
Tingkat Isolasi Neptune 251
Contoh transaksi 259
Klaster dan Instans 264
Instans DB primer 264
Instans replika baca 264
Pengukuran instans 265
Pemantauan instans 266
Penyimpanan, keandalan, dan ketersediaan 267
I/O — Penyimpanan yang dioptimalkan 267
Alokasi 268
Penagihan penyimpanan 268
Praktik terbaik penyimpanan 269
Keandalan dan ketersediaan tinggi 270
Koneksi Titik akhir 271
Titik akhir cluster 271
Titik akhir pembaca 272
Endpoint instans 273
Titik akhir kustom 273
Pertimbangan titik akhir 274

Bekerja dengan titik akhir khusus 275
Mode Lab 279
Menggunakan Mode Lab 279
Indeks OSGP 280
Pengumpulan sampah kamus 281
Semantik Transaksi 281
Dukungan datetime yang diperpanjang 282
StrictTimeoutValidation 282
QRCMemoryEstimasi Akurat 282
Mesin DFE Neptune 284
Mengontrol penggunaan DFE 284
Kueri yang dieksekusi oleh DFE 285
Statistik DFE 287
Batas ukuran 288
Status statistik 288
Nonaktifkan komputasi otomatis 290
Aktifkan kembali komputasi otomatis 291
Menghasilkan statistik secara manual 291
Pantau statistik 292
Autentikasi IAM 294
Hapus statistik 294
Kesalahan umum 295
API ringkasan grafik 297
Mengambil ringkasan grafik 298
`mode`Parameternya 298
Ringkasan grafik properti 299
Ringkasan grafik RDF 301
Contoh ringkasan PG 302
Contoh ringkasan RDF 305
IAM dan ringkasan grafik 309
Kesalahan ringkasan grafik umum 310
Konektivitas JDBC 313
Memulai 313
Menggunakan Tableau 314
Pemecahan masalah 316
Pembaruan mesin Neptunus 318

Pengecualian dan percobaan ulang 319
Mengapa coba lagi logika penting 319
Risiko percobaan ulang tanpa pandang bulu 319
Mengembangkan strategi coba lagi yang praktis 320
Contoh alur perencanaan 320
Pemantauan dan observabilitas 322
Keamanan 324
Peningkatan sistem operasi Amazon Neptunus 325
Meminimalkan downtime untuk upgrade OS yang memerlukan restart 325
Menerapkan upgrade OS ke instans DB Neptunus 326
Melindungi data dalam database Neptunus 328
Mengamankan database Neptunus dengan Amazon VPC 330
Mengenkripsi terhubung ke database Neptunus 330
Mengenkripsi data saat Istirahat 332
Autentikasi dengan IAM 335
Peran yang berbeda 336
Menggunakan Identitas 337
Mengaktifkan IAM 338
Menghubungkan dengan otentikasi IAM 339
Prasyarat 340
Menghubungkan dari baris perintah 342
Menghubungkan dari Gremlin Console 344
Menghubungkan menggunakan Gremlin Java 346
Terhubung dengan SPARQL Java 350
Berhubungan dengan SPARQL dan Node.js 354
Menghubungkan dengan Python 357
Python Gremlin 368
Gremlin Javascript 369
Gremlin Pergi 371
Gremlin.NET 376
Menggunakan kebijakan IAM 378
Kebijakan Berbasis Identitas 379
Kebijakan Kontrol Layanan (SCP) 379
Akses Konsol Neptune 380
Melampirkan Kebijakan 380
Jenis kebijakan IAM 381

Menggunakan tombol kondisi 381
Dukungan fitur IAM 382
Keterbatasan Kebijakan IAM 383
Menggunakan kebijakan terkelola 383
Menggunakan tombol kondisi 402
Pernyataan kebijakan administratif IAM 403
Pernyataan kebijakan akses data kustom 428
Peran Tertaut Layanan 448
Izin Peran 448
Membuat Peran Tertaut Layanan 450
Mengedit Peran Tertaut Layanan 451
Menghapus Peran Tertaut Layanan 451
Kredensial sementara 453
Dapatkan Kredensi dengan AWS CLI 454
Menyiapkan Lambda 458
Menyiapkan Amazon EC2 459
Pembuatan Log dan Pemantauan 461
Antarmuka VPC endpoint (AWS PrivateLink) 462
Pertimbangan kepatuhan 462
Membangun penyebaran yang tangguh 462
Migrasi ke Neptune 464
Migrasi dari Neo4j 465
Informasi umum 465
Mempersiapkan migrasi 468
Penyediaan infrastruktur 474
Migrasi data 477
Migrasi aplikasi 483
Kompatibilitas Neptune 487
Cypher menulis ulang 492
Sumber daya migrasi 500
Migrasi dari TinkerPop 501
Migrasi dari RDF 502
Menggunakan AWS DMS untuk bermigrasi 503
Migrasi dari Blazegraph 505
Kompatibilitas Neptune 505
Penyediaan infrastruktur 506

Mengekspor data ... 507
Buat bucket Amazon S3. ... 509
Mengimpor data ... 510
Memuat data ... 512
Bulk Loader Neptune ... 512
IAM role dan Akses Amazon S3 ... 514
Format Data ... 523
Contoh Pemuatan ... 538
Mengoptimalkan pemuatan massal ... 544
Referensi Loader ... 546
Memuat data menggunakan DMS ... 576
GraphMappingConfig ... 577
Mereplikasi ke Neptune ... 581
Muat data menggunakan kueri ... 586
OpenCypher: neptune.read () ... 587
SPARQL: MUAT dan BONGKAR ... 587
Gremlin: io () langkah ... 587
Penyimpanan ... 588
Pengumpulan sampah kamus ... 588
ID tepi yang dihasilkan server sebaris ... 590
Mengajukan kueri ... 593
Antrean kueri ... 594
Menemukan berapa banyak kueri dalam antrean ... 594
Waktu kueri ... 594
Cache rencana kueri ... 595
Cara memaksa mengaktifkan atau menonaktifkan cache rencana kueri ... 595
Cara menentukan apakah rencana di-cache atau tidak ... 595
Penggusuran ... 596
Kondisi yang menyebabkan rencana tidak di-cache ... 596
queryId Kustom ... 597
Menggunakan Header HTTP ... 597
Menggunakan Petunjuk Kueri SPARQL ... 598
Menggunakan queryId untuk Memeriksa Status ... 598
Gremlin ... 598
Menginstal konsol Gremlin ... 601
HTTPS REST ... 607

Java ... 610
Python ... 622
.NET ... 625
Node.js ... 628
Go ... 630
AWS SDK ... 633
Petunjuk kueri ... 637
Status kueri ... 647
Pembatalan kueri ... 649
Sesi berbasis skrip Gremlin ... 649
Transaksi Gremlin ... 652
Menggunakan API Gremlin ... 654
Hasil kueri cache ... 655
Peningkatan yang efisien dari 3.6.x maju ... 663
Peningkatan efisien sebelum 3.6.x ... 671
`explain` Gremlin ... 685
Gremlin dan DFE ... 734
OpenCypher ... 737
Gremlin vs OpenCypher ... 738
Menggunakan OpenCypher ... 739
Titik akhir status ... 740
Titik akhir HTTPS ... 744
AWS SDK ... 750
Menggunakan protokol Bolt ... 754
Contoh parameter ... 775
Model Data ... 777
OpenCypher `explain` ... 778
Transaksi ... 797
Petunjuk kueri ... 805
Pembatasan ... 810
Pengecualian ... 810
Ekstensi ... 816
Data Spasial ... 845
SPARQL ... 858
RDF4Konsol J ... 859
RDF4J Meja Kerja ... 861

Java ... 863
API HTTPS ... 867
Petunjuk kueri ... 881
DESKRIPSIKAN dan grafik default ... 898
Status kueri ... 900
Pembatalan kueri ... 903
Protokol penyimpanan grafik ... 904
`explain` SPARQL ... 906
Ekstensi `SERVICE` SPARQL ... 938
Alat visualisasi ... 942
Penjelajah grafik ... 942
Graph-explorer di notebook ... 943
Graph-explorer di Fargate ... 943
Demo ... 946
Perangkat Lunak Tom Sawyer ... 947
Kecerdasan Cambridge ... 948
Grafis ... 949
metafak ... 950
Klien database grafik G.V () ... 951
Fitur utama ... 951
Ketersediaan dan pembaruan ... 953
Demonstrasi produk ... 953
Pelajari selengkapnya ... 953
Linkurious ... 953
Graph.Build ... 954
Graph.Build Studio: ... 954
Graf.Build Transformer: ... 954
Penulis Graph.Build: ... 954
Mengekspor data ... 956
ekspor neptunus ... 956
Layanan Neptune-Ekspor ... 957
Menginstal layanan. ... 957
Mengaktifkan akses ke Neptune ... 961
Aktifkan akses ke Neptune-Ekspor ... 961
Menjalankan tugas ekspor ... 962
utilitas neptune-ekspor ... 965

Prasyarat ... 965
Menjalankan neptune-ekspor ... 966
Contoh perintah ... 967
File yang diekspor ... 968
Parameter Ekspor ... 969
perintah ... 971
Keluaran3Path ... 971
JobSize ... 972
params ... 972
TambahanParams ... 973
params ... 974
Contoh penyaringan ... 986
Pemecahan Masalah ... 990
Kesalahan umum ... 992
Mengekspor hasil kueri Gremlin ke Amazon S3 ... 993
Contoh ... 994
Prasyarat ... 994
Format respons ... 996
Memberikan akses untuk fitur ekspor Amazon S3 Gremlin ... 999
Mengelola Neptune ... 1002
Solusi Neptunus Blue/Green ... 1004
Prasyarat Neptunus Blue/Green ... 1005
Gunakan CloudFormation untuk menjalankan solusi ... 1006
Memantau progres ... 1007
Memotong ke cluster yang diperbarui ... 1009
Pembersihan ... 1010
Praktik terbaik ... 1010
Pemecahan masalah ... 1010
Izin pengguna IAM ... 1012
Kebijakan peran terkait layanan ... 1012
Buat pengguna IAM baru ... 1013
Grup parameter ... 1015
Mengedit Grup Parameter ... 1017
Membuat grup parameter ... 1018
Parameter ... 1019
neptune_enable_audit_log ... 1020

neptune_enable_slow_query_log 1020
neptune_slow_query_log_threshold 1020
neptune_lab_mode 1021
neptune_query_timeout 1021
neptune_stream 1022
neptune_streams_expiry_days 1022
neptune_lookup_cache 1023
neptune_autoscaling_config 1023
neptune_ml_iam_role 1024
neptune_ml_titik akhir 1024
neptune_enable_inline_server_generated_edge_id 1024
neptune_dfe_query_engine 1024
neptune_query_timeout 1025
neptune_result_cache 1025
UndoLogPurgeConfig 1026
neptune_enforce_ssl 1026
Luncurkan menggunakan konsol 1027
Menghentikan dan memulai klaster 1034
Menghentikan dan memulai ikhtisar 1034
Menghentikan klaster 1035
Memulai klaster DB 1036
API reset cepat 1038
Menggunakan IAM-Auth 1041
Magic %db_reset 1041
Kesalahan Umum 1042
Menambahkan instance pembaca 1045
Membuat instance pembaca 1046
Memodifikasi Klaster DB 1048
Memodifikasi Instans 1049
Performa dan Penskalaan 1050
Penskalaan Penyimpanan 1050
Penskalaan Instans 1050
Penskalaan Baca 1050
Penskalaan otomatis 1051
Penskalaan otomatis dan tanpa server 1053
Mengaktifkan auto-scaling 1053

Hapus auto-scaling 1057
Pemeliharaan cluster 1058
Nomor versi 1058
Jenis rilis 1059
Rentang hidup versi mesin 1061
Mengelola pembaruan mesin 1063
Proses upgrade 1068
Upgrade ke 1.2.0.0 atau lebih tinggi 1069
Perbarui melalui CloudFormation 1071
1.2.0.1 ke 1.2.0.2 1072
1.1.1.0 ke 1.2.0.2, default 1075
1.1.1.0 ke 1.2.0.2, kustom 1076
1.1.1.0 hingga 1.2.0.2, dicampur 1079
Mengkloning Klaster DB 1083
Batasan 1085
Copy-on-Write Protokol 1085
Menghapus Basis Data Sumber 1087
Mengelola Instans 1088
Instans Burstable T3 1089
Memodifikasi Instans 1091
Mengganti nama Instans DB Neptune 1096
Mem-boot ulang instans DB 1098
Menghapus Instans DB 1100
Aliran Neptunus 1102
Menggunakan Streams 1104
Mengaktifkan Streams 1105
Menonaktifkan Streams 1105
Memanggil Streams API 1105
Respon Streams 1107
Pengecualian Streams 1110
Format Catatan Stream 1111
PG_JSON 1111
RDF-NQUADS 1114
Contoh Streams 1115
Contoh AT_SEQUENCE_NUMBER 1115
Contoh SEQUENCE_NUMBER 1116

Contoh TRIM_HORIZON 1117
Contoh TERBARU 1117
Contoh Kompresi 1119
Neptune-to-Neptune Pengaturan Replikasi 1120
Pilih CloudFormation template 1120
Tambahkan detail tumpukan 1122
Jalankan Templat 1126
Memperbarui poller aliran 1127
Pengaliran untuk pemulihan bencana 1127
Pengaturan replikasi 1128
Pertimbangan lainnya 1132
Pencarian teks lengkap Neptunus 1133
Pengaturan pencarian teks lengkap 1135
CloudFormation Template 1136
Database yang ada 1143
Memperbarui poller 1144
Menghentikan dan memulai poller 1145
OpenSearch Tanpa server 1146
Perubahan yang diperlukan untuk fungsi AWS Lambda poller 1147
Perubahan yang diperlukan untuk utilitas impor massal 1147
Perubahan yang diperlukan pada peran IAM yang digunakan untuk menanyakan Neptunus 1147
Meminta dengan kontrol akses berbutir halus 1148
Penggunaan sintaks Lucene 1149
Model data pencarian teks lengkap Neptunus 1150
Dokumen sampel SPARQL 1151
Dokumen sampel Gremlin 1152
Parameter pencarian teks lengkap 1154
Pengindeksan non-string 1158
Memperbarui tumpukan yang ada 1159
Tidak termasuk bidang 1161
Pemetaan tipe data 1164
Validasi tipe data 1165
Kueri Sampel 1171
Full-text-search eksekusi kueri 1173
Contoh permintaan pencarian teks lengkap SPARQL 1175

Kueri pencocokan 1175
prefix 1176
fuzzy 1176
term 1176
query_string 1177
simple_query_string 1177
urutkan berdasarkan bidang string 1177
urutkan berdasarkan bidang non-string 1177
urutkan berdasarkan ID 1178
urutkan berdasarkan label 1179
urutkan berdasarkan doc_type 1179
Sintaks Lucene 1179
Contoh permintaan pencarian teks lengkap Gremlin 1180
Dasar `match` 1181
`match` 1181
`fuzzy` 1181
`query_string`kabur 1182
`query_string`regex 1182
Kueri hibrida 1182
Contoh pencarian teks lengkap 1183
`query_string`, '+' dan '-' 1183
`query_string`, `AND` dan `OR` 1185
`term` 1185
`prefix` 1185
Sintaks Lucene 1186
TinkerPop Grafik modern 1187
Urutkan berdasarkan bidang string 1188
Urutkan berdasarkan bidang non-string 1188
Urutkan berdasarkan bidang ID 1188
Urutkan berdasarkan bidang label 1188
Urutkan berdasarkan `document_type` bidang 1189
Pemecahan masalah dan metrik 1189
Membaca pemecahan masalah 1190
Menulis pemecahan masalah 1190
Out-of-sync masalah 1191
AWS Lambda fungsi 1193

Koneksi Gremlin WebSocket 1193
Rekomendasi Lambda Gremlin 1194
Rekomendasi permintaan tulis 1195
Rekomendasi permintaan baca 1196
Latensi awal dingin 1196
Membuat fungsi Lambda 1197
Contoh fungsi Lambda 1200
Contoh Java 1201
JavaScript contoh 1206
Contoh Python 1210
Machine learning Neptune 1215
Kemampuan Neptune ML 1215
Pengaturan Neptunus Neptunus 1217
Pengaturan menggunakan CloudFormation 1218
Pengaturan manual 1222
Menggunakan AWS CLI 1231
Menggunakan Neptune ML 1235
Memulai alur kerja 1235
Menangani data yang berkembang 1238
Memperbarui artefak model 1238
Alur kerja model kustom 1240
Pemilihan instans 1241
Untuk pengolahan data 1241
Untuk pelatihan model dan transformasi model 1241
Untuk titik akhir inferensi 1242
Ekspor data 1243
Contoh Ekspor Neptunus 1243
Pengaturan Params 1244
AdditionalParams 1245
sasaran 1248
fitur 1254
Contoh 1263
Pemrosesan data 1278
Mengelola pemrosesan data 1278
Pemrosesan yang diperbarui 1278
Pengkodean fitur 1280

Mengedit file data pelatihan 1288
Pelatihan model 1299
Model dan pelatihan 1301
Mengkustomisasi hyperparameters 1304
Pelatihan praktik terbaik 1317
Transformasi model 1321
Inferensi inkremental 1321
Transformasi model untuk pekerjaan apa pun 1321
Artefak model 1323
Artefak untuk tugas yang berbeda 1323
Menghasilkan artefak baru 1323
Model kustom 1326
Ikhtisar model khusus 1327
Pengembangan model kustom 1330
Titik akhir inferensi 1335
Mengelola endpoint inferensi 1335
Kueri inferensi 1336
Kueri inferensi Gremlin 1337
Kueri inferensi SPARQL 1362
API Neptune ML 1369
Perintah pemrosesandata 1370
Perintah modeltraining 1376
Perintah modeltransform 1383
Perintah titik akhir 1389
Pengecualian 1394
Batas 1395
SageMaker Batas AI 1396
Pemantauan Neptune 1397
Status instans 1398
Output sampel 1401
Menggunakan CloudWatch 1402
Menggunakan Konsol Tersebut 1402
Menggunakan AWS CLI 1403
Menggunakan CloudWatch API 1403
Memantau performa instans 1404
Metrik Neptune 1405

Dimensi Neptune 1422
Log Audit dengan Neptune 1423
Mengaktifkan Log Audit 1424
Melihat Log Audit 1424
Detail Log Audit 1424
Log Neptunus CloudWatch 1425
Publikasikan Log ke CloudWatch Log (Konsol) 1426
Publikasikan log audit ke CloudWatch Log (CLI) 1426
Publikasikan log kueri lambat ke CloudWatch Log (CLI) 1427
Memantau Log Acara 1427
CloudWatch Log Notebook 1428
Log kueri lambat 1430
Melihat log di konsol 1430
File log kueri lambat 1431
`info`atribut modus 1431
`debug`atribut modus 1434
Contoh keluaran 1436
Mencatat Panggilan API Neptunus dengan AWS CloudTrail 1438
Informasi Neptunus di CloudTrail 1439
Memahami entri file log 1440
Notifikasi peristiwa 1442
Kategori dan pesan 1443
Berlangganan peristiwa 1456
Kelola langganan 1457
Menandai sumber daya Neptunus 1458
Ikhtisar penandaan 1459
Menandai di konsol 1461
Menandai dengan CLI 1462
Menandai dengan API 1463
Bekerja dengan ARNs 1464
Mencadangkan dan memulihkan 1469
Cadangan 1469
Toleransi Kesalahan 1470
Metrik pencadangan 1471
Memulihkan Data 1472
Jendela cadangan 1473

Membuat Snapshot 1475
Menggunakan Konsol Tersebut 1475
Memulihkan dari snapshot 1476
Pertimbangan pemulihan penting 1476
Memulihkan 1478
Menyalin Snapshot 1479
Batasan 1479
Retensi Salinan Snapshot 1480
Enkripsi 1480
Penyalinan Snapshot Lintas Wilayah 1481
Menyalin Snapshot pada Konsol 1481
Menyalin Snapshot dengan AWS CLI 1482
Berbagi Snapshot 1486
Snapshot Terenkripsi 1487
Pembagian 1490
Menghapus Snapshot 1493
Menggunakan Konsol Tersebut 1493
Menggunakan AWS CLI 1493
Menggunakan API Neptune 1493
Praktik terbaik 1494
Pedoman operasional dasar 1496
Keamanan 1497
Menghindari ukuran instans yang berbeda 1498
Hindari restart beban massal 1499
Jika Anda memiliki banyak predikat 1499
Menghindari transaksi yang berjalan lama 1499
Menggunakan Metrik 1500
Penyetelan Kueri 1501
Penyeimbangan beban 1501
Gunakan contoh sementara 1501
Mengubah ukuran instance 1502
Kesalahan terputus tugas 1502
Gremlin (Umum) 1503
Konfigurasi Detak Jantung 1504
Optimalkan kueri upsert 1506
Perbedaan eksekusi GLV 1506

Penulisan Multithreaded 1507
Pemangkasan Catatan 1508
`datetime( )` 1508
Tanggal dan Waktu Asli 1509
Gremlin (klien Java) 1511
Gunakan kembali objek klien 1512
Klien Terpisah untuk Baca dan Tulis 1512
Beberapa titik akhir replika 1512
Tutup klien setelah selesai 1513
Koneksi baru setelah failover 1513
Set maxInProcess PerConnection = maxSimultaneousUsage PerConnection 1513
Kirim kueri sebagai bytecode 1514
Sepenuhnya mengkonsumsi hasil kueri 1515
Massal menambahkan simpul dan tepi 1516
Nonaktifkan cache DNS JVM 1517
Batas waktu per kueri 1517
Menangani `TimeoutException` 1518
OpenCypher dan Bolt 1520
Sambungkan kembali setelah failover 1521
Gunakan kembali objek Driver 1521
Penanganan koneksi Lambda 1521
Lebih suka tepi yang diarahkan 1521
Tidak ada kueri transaksi bersamaan 1522
Tutup objek pengemudi 1523
Gunakan mode transaksi eksplisit 1523
Logika coba lagi 1526
Menggunakan klausa SET 1529
Gunakan kueri berparameter 1530
CLASE UNWIND 1531
Ekspresi Jalur Panjang Variabel (VLP) 1532
Gunakan nama hubungan granular 1533
Tentukan label tepi 1534
Hindari klausa WITH 1535
Filter terbatas 1535
Periksa properti secara eksplisit 1536
Hindari jalur bernama 1537

Hindari KUMPULKAN (DISTINCT ()) 1537
Lebih suka fungsi properti 1538
Perhitungan statis 1539
Masukan batching 1539
Lebih suka kustom IDs 1540
Hindari perhitungan ~id 1541
Memperbarui/Menggabungkan beberapa node 1542
SPARQL 1543
Mengajukan Kueri Semua Grafik Bernama 1543
Tentukan Grafik Bernama untuk Dimuat 1544
FILTER vs. VALUES 1544
Batas Neptune 1547
Wilayah 1547
Wilayah Tiongkok 1548
Asia Pasifik (Hyderabad) 1548
Ukuran volume cluster 1548
Ukuran instans 1548
Per akun 1549
VPC Diperlukan 1549
SSL Diperlukan 1549
Availability Zone dan Subnet Grup 1550
HTTP permintaan payload 1550
Gremlin 1550
Tidak ada karakter nol 1550
SPARQL UPDATE LOAD 1550
Autentikasi dan Akses 1551
WebSockets Batas 1551
Properti dan label 1554
Pemuatan massal 1555
Integrasi Neptunus 1556
Alat dan utilitas 1558
GraphQL utilitas 1558
Instalasi dan pengaturan 1560
Menggunakan data yang ada 1560
Menggunakan skema tanpa arahan 1561
Bekerja dengan arahan 1566

Argumen baris perintah 1571
Nodestream 1575
Kesalahan Neptune 1577
Instans DB tidak cukup tersedia 1577
Kode Kesalahan Mesin 1577
Format Kesalahan 1578
Kesalahan Kueri 1578
Kesalahan IAM 1583
Kesalahan API 1585
Kesalahan Loader 1588
Rilis mesin 1591
Perencanaan rentang hidup versi mesin 1594
Rilis: 1.4.7.0 (2026-03-03) 1595
Fitur 1596
Perbaikan 1596
Perbaikan Cacat 1596
Versi Kueri Bahasa yang Didukung 1597
Jalur peningkatan 1597
Meningkatkan 1597
Rilis: 1.4.6.3 (2025-12-18) 1599
Cacat diperbaiki 1599
Versi Kueri Bahasa yang Didukung 1600
Jalur peningkatan 1600
Meningkatkan 1600
Rilis: 1.4.6.2 (2025-11-18) 1603
Cacat diperbaiki 1603
Versi Kueri Bahasa yang Didukung 1603
Jalur peningkatan 1604
Meningkatkan 1604
Rilis: 1.4.6.1 (2025-09-18) 1606
Cacat diperbaiki 1606
Versi Kueri Bahasa yang Didukung 1606
Jalur peningkatan 1607
Meningkatkan 1607
Rilis: 1.4.6.0 (2025-09-02) 1609
Fitur baru 1609

Perbaikan 1609
Cacat diperbaiki 1610
Versi Kueri Bahasa yang Didukung 1610
Jalur peningkatan 1611
Meningkatkan 1611
Rilis: 1.4.5.1 (2025-06-30) 1613
Cacat diperbaiki 1613
Versi Kueri Bahasa yang Didukung 1614
Jalur peningkatan 1614
Meningkatkan 1614
Rilis: 1.4.5.0 (2025-04-09) 1616
Fitur baru 1616
Perbaikan 1617
Cacat diperbaiki 1618
Versi Kueri Bahasa yang Didukung 1619
Jalur peningkatan 1620
Meningkatkan 1620
Rilis: 1.4.4.0 (2025-02-24) 1622
Cacat diperbaiki 1622
Versi Kueri Bahasa yang Didukung 1623
Jalur peningkatan 1623
Meningkatkan 1623
Rilis: 1.4.3.0 (2025-01-21) 1625
Fitur baru 1626
Perbaikan 1627
Cacat diperbaiki 1627
Versi Kueri Bahasa yang Didukung 1628
Jalur peningkatan 1628
Meningkatkan 1628
Rilis: 1.4.2.0 (2024-12-19) 1630
Cacat diperbaiki 1631
Versi Kueri Bahasa yang Didukung 1631
Jalur peningkatan 1631
Meningkatkan 1632
Rilis: 1.4.1.0 (2024-11-21) 1634
Fitur baru 1634

Perbaikan 1635
Cacat diperbaiki 1636
Versi Kueri Bahasa yang Didukung 1637
Jalur peningkatan 1637
Meningkatkan 1637
Rilis: 1.4.0.0 (2024-11-06) 1639
Fitur baru 1640
Perbaikan 1641
Cacat diperbaiki 1642
Versi Kueri Bahasa yang Didukung 1643
Jalur peningkatan 1643
Meningkatkan 1643
Rilis: 1.3.4.0 (2024-10-01) 1645
Perbaikan 1646
Cacat diperbaiki 1647
Versi Kueri Bahasa yang Didukung 1648
Jalur peningkatan 1648
Meningkatkan 1648
Rilis: 1.3.3.0 (2024-08-05) 1651
Cacat diperbaiki 1651
Mengurangi masalah dalam rilis 1.3.3.0 1652
Versi Kueri Bahasa yang Didukung 1653
Jalur peningkatan 1653
Meningkatkan 1653
Rilis: 1.3.2.1 (2024-06-20) 1655
Cacat diperbaiki 1656
Perubahan 1.3.2.1 dibawa dari 1.3.2.0 1658
Jalur peningkatan 1662
Meningkatkan 1663
Rilis: 1.3.2.0 (2024-06-10) 1665
Perbaikan 1666
Cacat diperbaiki 1667
Mengurangi masalah dalam rilis 1.3.2.0 1669
Versi Kueri Bahasa yang Didukung 1671
Jalur peningkatan 1671
Meningkatkan 1671

Rilis: 1.3.1.0 (2024-03-06) 1674
Perbaikan 1674
Cacat diperbaiki 1674
Versi Kueri Bahasa yang Didukung 1675
Jalur peningkatan 1676
Meningkatkan 1676
Rilis: 1.3.0.0 (2023-11-15) 1678
Fitur Baru 1678
Perbaikan 1679
Perbaikan Cacat 1681
Versi Kueri Bahasa yang Didukung 1682
Jalur Peningkatan 1682
Meningkatkan 1682
Rilis: 1.2.1.2 (2024-08-05) 1684
Perbaikan Cacat 1686
Versi Kueri Bahasa yang Didukung 1686
Jalur Peningkatan 1686
Meningkatkan 1686
Rilis: 1.2.1.1 (2024-03-11) 1689
Perbaikan 1690
Perbaikan Cacat 1690
Versi Kueri Bahasa yang Didukung 1691
Jalur Peningkatan 1691
Meningkatkan 1691
Rilis: 1.2.1.0 (2023-03-08) 1693
Rilis Patch 1694
Fitur Baru 1695
Perbaikan 1696
Perbaikan Cacat 1697
Versi Kueri Bahasa yang Didukung 1698
Jalur Peningkatan 1698
Meningkatkan 1699
Rilis: 1.2.1.0.R7 (2023-10-06) 1701
Rilis: 1.2.1.0.R6 (2023-09-12) 1704
Rilis: 1.2.1.0.R5 (2023-09-02) 1708
Rilis: 1.2.1.0.R4 (2023-08-10) 1712

Rilis: 1.2.1.0.R3 (2023-06-13) 1716
Rilis: 1.2.1.0.R2 (2023-05-02) 1722
Rilis: 1.2.0.2 (2022-11-20) 1726
Rilis Patch 1727
Fitur Baru 1727
Perbaikan 1728
Versi Kueri Bahasa yang Didukung 1728
Jalur Peningkatan 1728
Meningkatkan 1728
Rilis: 1.2.0.2.R6 (2023-09-12) 1730
Rilis: 1.2.0.2.R5 (2023-08-16) 1734
Rilis: 1.2.0.2.R4 (2023-05-08) 1737
Rilis: 1.2.0.2.R3 (2023-03-27) 1741
Rilis: 1.2.0.2.R2 (2022-12-15) 1745
Rilis: 1.2.0.1 (2022-10-26) 1750
Rilis Patch 1751
Fitur Baru 1751
Perbaikan 1751
Perbaikan Cacat 1752
Versi Kueri Bahasa yang Didukung 1752
Jalur Peningkatan 1753
Meningkatkan 1753
Rilis Pemeliharaan: 1.2.0.1.R3 (2023-09-27) 1755
Rilis Pemeliharaan: 1.2.0.1.R2 (2022-12-13) 1759
Rilis: 1.2.0.0 (2022-07-21) 1763
Rilis Patch 1764
Fitur Baru 1764
Perbaikan 1765
Perbaikan Cacat 1766
Versi Kueri Bahasa yang Didukung 1768
Jalur Peningkatan 1768
Meningkatkan 1769
Rilis: 1.2.0.0.R4 (2023-09-29) 1771
Rilis: 1.2.0.0.R3 (2022-12-15) 1776
Rilis: 1.2.0.0.R2 (2022-10-14) 1781
Rilis: 1.1.1.0 (2022-04-19) 1787

Rilis Patch 1788
Fitur Baru 1788
Perbaikan 1789
Perbaikan Cacat 1790
Versi Kueri Bahasa yang Didukung 1791
Jalur peningkatan 1791
Meningkatkan 1792
Rilis: 1.1.1.0.R7 (2023-01-23) 1795
Rilis: 1.1.1.0.R6 (2022-09-23) 1800
Rilis: 1.1.1.0.R5 (2022-07-21) 1805
Rilis: 1.1.1.0.R4 (2022-06-23) 1810
Rilis: 1.1.1.0.R3 (2022-06-07) 1815
Rilis pemeliharaan: 1.1.1.0.R2 (2022-05-16) 1820
Rilis: 1.1.0.0 (2021-11-19) 1825
Rilis patch 1826
Fitur Baru 1826
Perbaikan 1827
Perbaikan Cacat 1828
Versi Kueri Bahasa yang Didukung 1829
Jalur peningkatan 1829
Meningkatkan 1829
Rilis pemeliharaan: 1.1.0.0.R3 (2022-12-23) 1831
Rilis pemeliharaan: 1.1.0.0.R2 (2022-05-16) 1836
Rilis: 1.0.5.1 (2021-10-01) 1840
Rilis Patch 1841
Fitur Baru 1841
Perbaikan 1841
Perbaikan Cacat 1842
Versi Kueri Bahasa yang Didukung 1842
Jalur peningkatan 1842
Meningkatkan 1842
Rilis pemeliharaan: 1.0.5.1.R4 (2022-05-16) 1844
Rilis: 1.0.5.1.R3 (2022-01-13) 1847
Rilis: 1.0.5.1.R2 (2021-10-26) 1849
Rilis: 1.0.5.0 (2021-07-27) 1852
Rilis Patch 1852

Fitur Baru 1852
Perbaikan 1853
Perbaikan Cacat 1854
Versi Kueri Bahasa yang Didukung 1854
Jalur peningkatan 1854
Meningkatkan 1854
Rilis pemeliharaan: 1.0.5.0.R5 (2022-05-16) 1856
Rilis: 1.0.5.0.R3 (2021-09-15) 1859
Rilis: 1.0.5.0.R2 (2021-08-16) 1862
Rilis: 1.0.4.2 (2021-06-01) 1865
Rilis: 1.0.4.2.R5 (2021-08-16) 1865
Rilis: 1.0.4.2.R4 (2021-07-23) 1866
Rilis: 1.0.4.2.R3 (2021-06-28) 1867
Rilis: 1.0.4.2.R2 (2021-06-01) 1868
Rilis: 1.0.4.2.R1 (2021-05-27) 1872
Rilis: 1.0.4.1 (2020-12-08) 1872
Rilis Patch 1872
Fitur Baru 1873
Perbaikan 1873
Perbaikan Cacat 1873
Versi Query Bahasa yang Didukung 1874
Jalur Peningkatan 1874
Meningkatkan 1874
Rilis: 1.0.4.1.R1.1 (2021-03-22) 1876
Rilis: 1.0.4.1.R2 (2021-02-24) 1879
Rilis: 1.0.4.0 (2020-10-12) 1884
Rilis Patch 1884
Fitur Baru 1885
Perbaikan 1885
Perbaikan Cacat 1886
Versi Bahasa Kueri yang Didukung 1886
Jalur Peningkatan 1887
Meningkatkan 1887
Rilis: 1.0.4.0.R2 (2021-02-24) 1889
Rilis: 1.0.3.0 (2020-08-03) 1892
Rilis Patch 1892

Fitur Baru 1892
Perbaikan 1893
Perbaikan Cacat 1893
Versi Kueri Bahasa yang Didukung 1894
Jalur Peningkatan 1894
Meningkatkan 1894
Rilis: 1.0.3.0.R3 (2021-02-19) 1896
Rilis: 1.0.3.0.R2 (2020-10-12) 1899
Rilis: 1.0.2.2 (2020-03-09) 1902
Rilis patch 1902
Perbaikan 1902
Perbaikan Cacat 1903
Versi Bahasa Kueri yang Didukung 1903
Jalur Peningkatan 1904
Meningkatkan 1904
Rilis: 1.0.2.2.R6 (2021-02-19) 1906
Rilis: 1.0.2.2.R5 (2020-10-12) 1908
Rilis: 1.0.2.2.R4 (2020-07-23) 1912
Rilis: 1.0.2.2.R3 (2020-07-22) 1915
Rilis: 1.0.2.2.R2 (2020-04-02) 1915
Rilis: 1.0.2.1 (2019-11-22) 1918
Rilis patch 1918
Fitur Baru 1918
Perbaikan 1919
Perbaikan Cacat 1919
Versi Bahasa Kueri yang Didukung 1920
Jalur peningkatan 1920
Meningkatkan 1920
Rilis: 1.0.2.1.R6 (2020-04-22) 1922
Rilis: 1.0.2.1.R5 (2020-04-22) 1925
Rilis: 1.0.2.1.R4 (2019-12-20) 1925
Rilis: 1.0.2.1.R3 (2019-12-12) 1928
Rilis: 1.0.2.1.R2 (2019-11-25) 1931
Rilis: 1.0.2.0 (2019-11-08) 1933
PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN 1933
Rilis Patch 1933

Fitur Baru 1934
Versi Bahasa Kueri yang Didukung 1934
Jalur Peningkatan 1934
Meningkatkan 1934
Rilis: 1.0.2.0.R3 (2020-05-05) 1936
Rilis: 1.0.2.0.R2 (2019-11-21) 1939
Rilis: 1.0.1.2 (2020-06-10) 1942
PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN 1942
Perbaikan 1942
Perbaikan Cacat 1943
Versi Bahasa Kueri yang Didukung 1943
Rilis: 1.0.1.1 (2020-06-26) 1943
PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN 1943
Perbaikan Cacat 1943
Versi Bahasa Kueri yang Didukung 1943
Rilis: 1.0.1.0 (2019-07-02) 1944
PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN 1944
Rilis 1.0.1.0.200502.0 (2019-10-31) 1944
Rilis 1.0.1.0.200463.0 (2019-10-15) 1944
Rilis 1.0.1.0.200457.0 (2019-09-19) 1946
Rilis 1.0.1.0.200369.0 (2019-08-13) 1947
Rilis 1.0.1.0.200366.0 (2019-07-26) 1947
Rilis 1.0.1.0.200348.0 (2019-07-02) 1949
Rilis Sebelumnya 1950
Menggunakan Neptunus APIs 1963
Tindakan IAM bersama 1963
Referensi API Manajemen 1970
Klaster 1977
Buat DBCluster 1977
Hapus DBCluster 1989
Memodifikasi DBCluster 1995
Mulai DBCluster 2006
Berhenti DBCluster 2011
AddRoleToDBCluster 2017
RemoveRoleFromDBCluster 2018
Kegagalan DBCluster 2019

PromoteReadReplicaDBCluster 2025
Jelaskan DBClusters 2031
_____ 2033
DBCluster 2033
DBClusterAnggota 2038
DBClusterPeran 2039
CloudwatchLogsExportConfiguration 2040
PendingCloudwatchLogsExports 2040
ClusterPendingModifiedValues 2041
Database global 2042
CreateGlobalCluster 2042
DeleteGlobalCluster 2045
ModifyGlobalCluster 2047
DescribeGlobalClusters 2050
FailoverGlobalCluster 2051
RemoveFromGlobalCluster 2053
_____ 2055
GlobalCluster 2055
GlobalClusterMember 2057
Instans 2057
Buat DBInstance 2058
Hapus DBInstance 2070
Memodifikasi DBInstance 2076
Reboot DBInstance 2088
Jelaskan DBInstances 2094
DescribeOrderableDBInstancePilihan 2096
DescribeValidDBInstanceModifikasi 2097
_____ 2098
DBInstance 2098
DBInstanceStatusInfo 2103
Opsi yang Dapat Dipesan DBInstance 2104
PendingModifiedValues 2106
ValidStorageOptions 2107
Valid DBInstance ModificationsMessage 2108
Parameter 2108
Salin DBParameter Grup 2109

Salin DBCluster ParameterGroup 2111
Buat DBParameter Grup 2112
Buat DBCluster ParameterGroup 2115
Hapus DBParameter Grup 2117
Hapus DBCluster ParameterGroup 2118
Ubah DBParameter Grup 2118
Memodifikasi DBCluster ParameterGroup 2120
Setel Ulang DBParameter Grup 2122
Setel ulang DBCluster ParameterGroup 2123
Jelaskan DBParameters 2124
Jelaskan DBParameter Grup 2126
Jelaskan DBCluster Parameter 2127
Jelaskan DBCluster ParameterGroups 2128
DescribeEngineDefaultParameters 2130
DescribeEngineDefaultClusterParameters 2131
_____ 2132
Parameter 2132
DBParameterKelompok 2133
DBClusterParameterGroup 2134
DBParameterGroupStatus 2135
Subnet 2135
Buat DBSubnet Grup 2136
Hapus DBSubnet Grup 2137
Ubah DBSubnet Grup 2138
Jelaskan DBSubnet Grup 2140
_____ 2141
Subnet 2141
DBSubnetKelompok 2142
Snapshot 2143
Buat DBCluster Snapshot 2143
Hapus DBCluster Snapshot 2146
Salin DBCluster Snapshot 2149
Memodifikasi DBCluster SnapshotAttribute 2154
Kembalikan DBCluster FromSnapshot 2156
Kembalikan DBCluster ToPointInTime 2165
Jelaskan DBCluster Snapshots 2175

Jelaskan DBCluster SnapshotAttributes 2178
_____ 2179
DBClusterCuplikan 2179
DBClusterSnapshotAttribute 2181
DBClusterSnapshotAttributesResult 2182
Peristiwa 2183
CreateEventSubscription 2183
DeleteEventSubscription 2186
ModifyEventSubscription 2188
DescribeEventSubscriptions 2191
AddSourceIdentifierToSubscription 2192
RemoveSourceIdentifierFromSubscription 2194
DescribeEvents 2195
DescribeEventCategories 2198
_____ 2198
Peristiwa 2198
EventCategoriesMap 2199
EventSubscription 2199
Lainnya 2201
AddTagsToResource 2201
ListTagsForResource 2202
RemoveTagsFromResource 2203
ApplyPendingMaintenanceAction 2204
DescribePendingMaintenanceActions 2205
Jelaskan DBEngine Versi 2207
_____ 2209
DBEngineVersi 2209
EngineDefaults 2210
PendingMaintenanceAction 2210
ResourcePendingMaintenanceActions 2211
UpgradeTarget 2212
Tag 2213
DataType 2213
AvailabilityZone 2214
DBSecurityGroupMembership 2214
DomainMembership 2214

DoubleRange 2215
Titik Akhir 2215
Filter 2215
Kisaran 2216
ServerlessV2 ScalingConfiguration 2216
ServerlessV2 ScalingConfigurationInfo 2217
Zona waktu 2217
VpcSecurityGroupMembership 2218
Kesalahan API 2218
AuthorizationAlreadyExistsFault 2221
AuthorizationNotFoundFault 2221
AuthorizationQuotaExceededFault 2221
CertificateNotFoundFault 2222
DBClusterAlreadyExistsFault 2222
DBClusterNotFoundFault 2222
DBClusterParameterGroupNotFoundFault 2223
DBClusterQuotaExceededFault 2223
DBClusterRoleAlreadyExistsFault 2223
DBClusterRoleNotFoundFault 2224
DBClusterRoleQuotaExceededFault 2224
DBClusterSnapshotAlreadyExistsFault 2224
DBClusterSnapshotNotFoundFault 2224
DBInstanceAlreadyExistsFault 2225
DBInstanceNotFoundFault 2225
DBLogFileNotFoundFault 2225
DBParameterGroupAlreadyExistsFault 2226
DBParameterGroupNotFoundFault 2226
DBParameterGroupQuotaExceededFault 2226
DBSecurityGroupAlreadyExistsFault 2227
DBSecurityGroupNotFoundFault 2227
DBSecurityGroupNotSupportedFault 2227
DBSecurityGroupQuotaExceededFault 2227
DBSnapshotAlreadyExistsFault 2228
DBSnapshotNotFoundFault 2228
DBSubnetGroupAlreadyExistsFault 2228
DBSubnetGroupDoesNotCoverEnoughAZs 2229

DBSubnetGroupNotAllowedFault 2229
DBSubnetGroupNotFoundFault 2229
DBSubnetGroupQuotaExceededFault 2230
DBSubnetQuotaExceededFault 2230
DBUpgradeDependencyFailureFault 2230
DomainNotFoundFault 2230
EventSubscriptionQuotaExceededFault 2231
GlobalClusterAlreadyExistsFault 2231
GlobalClusterNotFoundFault 2231
GlobalClusterQuotaExceededFault 2232
InstanceQuotaExceededFault 2232
Tidak cukup DBCluster CapacityFault 2232
Tidak cukup DBInstance CapacityFault 2233
InsufficientStorageClusterCapacityFault 2233
Tidak valid DBCluster EndpointStateFault 2233
Tidak valid DBCluster SnapshotStateFault 2233
Tidak valid DBCluster StateFault 2234
Tidak valid DBInstance StateFault 2234
Tidak valid DBParameter GroupStateFault 2234
Tidak valid DBSecurity GroupStateFault 2235
Tidak valid DBSnapshot StateFault 2235
Tidak valid DBSubnet GroupFault 2235
Tidak valid DBSubnet GroupStateFault 2236
Tidak valid DBSubnet StateFault 2236
InvalidEventSubscriptionStateFault 2236
InvalidGlobalClusterStateFault 2237
InvalidOptionGroupStateFault 2237
InvalidRestoreFault 2237
InvalidSubnet 2237
Tidak valid VPCNetwork StateFault 2238
KMSKeyNotAccessibleFault 2238
OptionGroupNotFoundFault 2238
PointInTimeRestoreNotEnabledFault 2239
ProvisionedIopsNotAvailableInAZFault 2239
ResourceNotFoundFault 2239
SNSInvalidTopicFault 2240

SNSNoAuthorizationFault 2240
SNSTopicArnNotFoundFault 2240
SharedSnapshotQuotaExceededFault 2241
SnapshotQuotaExceededFault 2241
SourceNotFoundFault 2241
StorageQuotaExceededFault 2241
StorageTypeNotSupportedFault 2242
SubnetAlreadyInUse 2242
SubscriptionAlreadyExistFault 2242
SubscriptionCategoryNotFoundFault 2243
SubscriptionNotFoundFault 2243
Referensi API data 2244
Umum 2248
GetEngineStatus 2248
ExecuteFastReset 2250
_____ 2252
QueryLanguageVersion 2252
FastResetToken 2252
Mengajukan kueri 2253
ExecuteGremlinQuery 2253
ExecuteGremlinExplainQuery 2256
ExecuteGremlinProfileQuery 2257
ListGremlinQueries 2259
GetGremlinQueryStatus 2261
CancelGremlinQuery 2262
_____ 2263
ExecuteOpenCypherQuery 2263
ExecuteOpenCypherExplainQuery 2265
ListOpenCypherQueries 2267
GetOpenCypherQueryStatus 2269
CancelOpenCypherQuery 2270
_____ 2272
QueryEvalStats 2272
GremlinQueryStatus 2272
GremlinQueryStatusAttributes 2273
Pemuat massal 2273

StartLoaderJob ... 2273
GetLoaderJobStatus ... 2280
ListLoaderJobs ... 2283
CancelLoaderJob ... 2284
_____ ... 2286
LoaderIdResult ... 2286
Pengaliran ... 2286
GetPropertygraphStream ... 2286
_____ ... 2289
PropertygraphRecord ... 2289
PropertygraphData ... 2290
Statistik ... 2291
GetPropertygraphStatistics ... 2292
ManagePropertygraphStatistics ... 2293
DeletePropertygraphStatistics ... 2294
GetPropertygraphSummary ... 2295
_____ ... 2297
Statistik ... 2297
StatisticsSummary ... 2298
DeleteStatisticsValueMap ... 2298
RefreshStatisticsIdMap ... 2298
NodeStructure ... 2299
EdgeStructure ... 2299
SubjectStructure ... 2299
PropertygraphSummaryValueMap ... 2300
PropertygraphSummary ... 2300
Pemrosesan data ML ... 2302
Mulai MLData ProcessingJob ... 2302
Daftar MLData ProcessingJobs ... 2305
Dapatkan MLData ProcessingJob ... 2306
Batalkan MLData ProcessingJob ... 2308
_____ ... 2309
MlResourceDefinition ... 2309
MlConfigDefinition ... 2310
Pelatihan model ML ... 2310
Mulai MLModel TrainingJob ... 2311

Daftar MLModel TrainingJobs 2314
Dapatkan MLModel TrainingJob 2315
Batalkan MLModel TrainingJob 2317
_____ 2318
CustomModelTrainingParameters 2318
Transformasi model ML 2319
Mulai MLModel TransformJob 2319
Daftar MLModel TransformJobs 2322
Dapatkan MLModel TransformJob 2324
Batalkan MLModel TransformJob 2325
_____ 2326
CustomModelTransformParameters 2326
Titik akhir Inferensi ML 2327
Buat MLEndpoint 2327
Daftar MLEndpoints 2330
Dapatkan MLEndpoint 2331
Hapus MLEndpoint 2332
Pengecualian 2333
AccessDeniedException 2335
BadRequestException 2335
BulkLoadIdNotFoundException 2336
CancelledByUserException 2336
ClientTimeoutException 2336
ConcurrentModificationException 2337
ConstraintViolationException 2337
ExpiredStreamException 2338
FailureByQueryException 2338
IllegalArgumentException 2338
InternalFailureException 2339
InvalidArgumentException 2339
InvalidNumericDataException 2340
InvalidParameterException 2340
LoadUrlAccessDeniedException 2340
MalformedQueryException 2341
MemoryLimitExceededException 2341
MethodNotAllowedException 2342

MissingParameterException 2342
MLResourceNotFoundException 2343
ParsingException 2343
PreconditionsFailedException 2343
QueryLimitExceededException 2344
QueryLimitException 2344
QueryTooLargeException 2345
ReadOnlyViolationException 2345
S3Exception 2345
ServerShutdownException 2346
StatisticsNotAvailableException 2346
StreamRecordsNotFoundException 2347
ThrottlingException 2347
TimeLimitExceededException 2348
TooManyRequestsException 2348
UnsupportedOperationException 2348
UnloadUrlAccessDeniedException 2349
..... mmcccl

Apa itu Amazon Neptune?

Amazon Neptune adalah layanan basis data grafik yang cepat, andal, terkelola penuh yang memudahkan membangun dan menjalankan aplikasi yang bekerja dengan set data yang sangat terhubung. Inti dari Neptune adalah mesin basis data grafik berperforma tinggi yang dibuat khusus. Mesin ini dioptimalkan untuk menyimpan miliaran hubungan dan membuat kueri grafik dengan latensi milidetik. Neptunus mendukung bahasa kueri grafik properti populer TinkerPop Apache Gremlin dan OpenCypher Neo4j, dan bahasa kueri RDF W3C, SPARQL. Ini memungkinkan Anda untuk membangun kueri yang secara efisien menavigasi kumpulan data yang sangat terhubung. Neptune mendukung kasus penggunaan grafik seperti mesin rekomendasi, deteksi penipuan, grafik pengetahuan, penemuan obat, dan keamanan jaringan.

Basis data Neptunus sangat tersedia, dengan replika point-in-time baca, pemulihan, pencadangan berkelanjutan ke Amazon S3, dan replikasi di seluruh Availability Zone. Neptune menyediakan fitur keamanan data, dengan dukungan enkripsi saat istirahat dan dalam transit. Neptune terkelola penuh, sehingga Anda tidak perlu lagi khawatir tentang tugas-tugas manajemen basis data seperti penyediaan perangkat keras, patch perangkat lunak, setup, konfigurasi, atau backup.

Neptunus Analytics adalah mesin database analitik yang melengkapi database Neptunus dan yang dapat dengan cepat menganalisis sejumlah besar data grafik dalam memori untuk mendapatkan wawasan dan menemukan tren. Neptunus Analytics adalah solusi untuk menganalisis database grafik atau kumpulan data grafik yang ada dengan cepat yang disimpan di danau data. Ini menggunakan algoritma analitik grafik populer dan kueri analitik latensi rendah.

Untuk mempelajari selengkapnya tentang menggunakan Amazon Neptune, kami sarankan Anda mulai dengan bagian berikut:

- Memulai dengan Amazon Neptunus
- Ikhtisar fitur Amazon Neptunus

Jika Anda baru dalam grafik, atau belum siap untuk berinvestasi di lingkungan produksi Neptune penuh, kunjungi topik Memulai dengan Neptunus untuk mengetahui cara menggunakan notebook Neptune Jupyter untuk belajar dan berkembang tanpa menimbulkan biaya.

Selain itu, sebelum Anda mulai merancang database, kami sarankan Anda berkonsultasi dengan Arsitektur AWS Referensi GitHub repositori untuk Menggunakan Database Grafik, di mana Anda

dapat menginformasikan pilihan Anda tentang model data grafik dan bahasa kueri, dan menelusuri contoh arsitektur penerapan referensi.

Komponen Layanan Utama

- Instans DB primer – Mendukung operasi baca dan tulis, serta melakukan semua modifikasi data pada volume klaster. Setiap klaster DB Neptune memiliki satu instans DB utama yang bertanggung jawab untuk menulis (yaitu, memuat atau memodifikasi) isi basis data grafik.
- Replika Neptune – Menghubungkan ke volume penyimpanan yang sama seperti instans DB utama dan mendukung operasi read-only. Setiap klaster DB Neptune dapat memiliki hingga 15 Replika Neptune di samping instans DB utama. Ini menyediakan ketersediaan tinggi dengan menemukan Replika Neptune di Availability Zone terpisah dan distribusi beban dari klien membaca.
- Volume klaster— Data Neptune disimpan dalam volume klaster, yang dirancang untuk keandalan dan ketersediaan tinggi. Volume cluster terdiri dari salinan data di beberapa Availability Zone dalam satu AWS Region. Karena data direplikasi secara otomatis di Availability Zone, data Anda sangat tahan lama dengan kemungkinan kehilangan data yang kecil.

Mendukung Grafik Terbuka APIs

Amazon Neptunus mendukung APIs grafik terbuka untuk grafik properti (Gremlin dan OpenCypher) dan grafik RDF (SPARQL). Hal ini memberikan kinerja tinggi untuk kedua model grafik ini dan bahasa kueri mereka. Anda dapat memilih model Grafik Properti (PG) dan mengakses grafik yang sama dengan bahasa kueri OpenCypher bahasa kueri and/or Gremlin. Jika Anda menggunakan model Resource Description Framework (RDF) standar W3C, Anda dapat mengakses grafik Anda menggunakan bahasa kueri SPARQL standar.

Sangat Aman

Neptune menyediakan beberapa tingkat keamanan untuk basis data Anda. Fitur keamanan mencakup isolasi jaringan menggunakan Amazon VPC, dan enkripsi saat istirahat menggunakan kunci yang Anda buat dan kontrol melalui AWS Key Management Service (AWS KMS). Pada instans Neptune terenkripsi, data dalam penyimpanan yang mendasari dienkripsi, seperti halnya backup otomatis, snapshot, dan replika dalam klaster yang sama.

Terkelola Penuh

Dengan Amazon Neptune, Anda tidak perlu lagi khawatir tentang tugas-tugas manajemen basis data seperti penyediaan perangkat keras, patch perangkat lunak, setup, konfigurasi, atau backup.

Anda dapat menggunakan Neptune untuk membuat aplikasi grafik interaktif canggih yang dapat melakukan kueri miliaran hubungan dalam milidetik. Kueri SQL untuk data yang sangat terhubung kompleks dan sulit untuk disetel demi kinerja. Dengan Neptunus, Anda dapat menggunakan bahasa kueri grafik populer Gremlin, OpenCypher, dan SPARQL untuk mengeksekusi kueri kuat yang mudah ditulis dan berkinerja baik pada data yang terhubung. Kemampuan ini secara signifikan mengurangi kompleksitas kode sehingga Anda dapat dengan cepat membuat aplikasi yang memproses hubungan.

Neptune dirancang untuk menawarkan ketersediaan lebih besar dari 99,99 persen. Neptune meningkatkan performa database dan ketersediaan dengan secara erat mengintegrasikan mesin database dengan lapisan penyimpanan virtualisasi berdukungan SSD yang dibangun untuk beban kerja basis data. Penyimpanan Neptune toleran akan kesalahan dan dapat memperbaiki diri. Kegagalan disk diperbaiki di latar belakang tanpa kehilangan ketersediaan basis data. Neptune secara otomatis mendeteksi crash pada basis data dan melakukan restart tanpa perlu pemulihan crash atau membangun kembali cache basis data. Jika seluruh instans gagal, Neptune secara otomatis melakukan failover ke salah satu hingga 15 replika bacanya.

Perubahan dan Pembaruan ke Amazon Neptune

Tabel berikut menjelaskan perubahan penting pada Amazon Neptune.

Perubahan	Deskripsi	Tanggal
Neptunus di wilayah Asia Pasifik (Hyderabad)	Amazon Neptunus sekarang tersedia di wilayah Asia Pasifik (Hyderabad) (). `ap-south-2`	10 Maret 2026
Versi mesin 1.4.7.0	Pada 2026-03-03, engine versi 1.4.7.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.4.7.0.	Maret 3, 2026
Versi mesin 1.4.6.3	Pada 2025-12-18, engine versi 1.4.6.3 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.4.6.3.	Desember 18, 2025
Versi mesin 1.4.6.2	Pada 2025-11-18, engine versi 1.4.6.2 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya	November 18, 2025

	tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.4.6.2.	
Versi mesin 1.4.6.1	Pada 2025-09-18, engine versi 1.4.6.1 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.4.6.1.	September 18, 2025
Versi mesin 1.4.6.0	Pada 2025-09-02, engine versi 1.4.6.0 umumnya digunakan . Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.4.6.0.	September 2, 2025
Titik akhir publik	Amazon Neptunus sekarang mendukung titik akhir publik. Untuk informasi selengkapnya lihat titik akhir Publik.	Agustus 27, 2025
Versi mesin 1.4.5.1	Pada 2025-06-30, engine versi 1.4.5.1 umumnya digunakan . Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkap nya tentang versi mesin ini, lihat Neptune Engine Release 1.4.5.1.	Juni 30, 2025

Neptunus diluncurkan di wilayah Asia Pasifik (Melbourne) dan Kanada Barat (Calgary)	Amazon Neptunus sekarang tersedia di wilayah Asia Pasifik (Melbourne) (`ap-southeast-4`) dan Kanada Barat (Calgary) (). `ca-west-1`	28 Mei 2025
Versi mesin 1.4.5.0	Pada 2025-04-09, engine versi 1.4.5.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.4.5.0.	April 9, 2025
Neptunus diluncurkan di wilayah Asia Pasifik (Malaysia)	Amazon Neptunus sekarang tersedia di wilayah Asia Pasifik (Malaysia) ()`ap-southeast-5` . Notebook terkelola tidak didukung untuk wilayah Asia Pasifik (Malaysia) saat ini, tetapi dapat diatur secara manual. Lihat Menyiapkan buku catatan Neptunus secara manual untuk informasi selengkapnya.	Februari 28, 2025

Versi mesin 1.4.4.0	Pada 2025-02-24, engine versi 1.4.4.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.4.4.0.	Februari 24, 2025
Versi mesin 1.4.3.0	Pada 2025-01-21, engine versi 1.4.3.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.4.3.0.	Januari 21, 2025
Versi mesin 1.4.2.0	Pada 2024-12-19, engine versi 1.4.2.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.4.2.0.	Desember 19, 2024

Versi mesin 1.4.1.0	Pada 2024-11-21, engine versi 1.4.1.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.4.1.0.	November 21, 2024
Versi mesin 1.4.0.0	Pada 2024-11-06, engine versi 1.4.0.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.4.0.0.	6 November 2024
Versi mesin 1.3.4.0	Pada 2024-10-01, engine versi 1.3.4.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.3.4.0.	Oktober 1, 2024

Versi mesin 1.3.3.0	Pada 2024-08-05, engine versi 1.3.3.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.3.3.0.	Agustus 5, 2024
Versi mesin 1.2.1.2	Pada 2024-08-05, engine versi 1.2.1.2 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.1.2.	Agustus 5, 2024
Versi mesin 1.3.2.1	Pada 2024-06-20, engine versi 1.3.2.1 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.3.2.1.	Juni 20, 2024

Versi mesin 1.3.2.0	Pada 2024-06-10, engine versi 1.3.2.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.3.2.0.	Juni 10, 2024
Versi mesin 1.2.1.1	Pada 2024-03-11, engine versi 1.2.1.1 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.1.1.	Maret 11, 2024
Versi mesin 1.3.1.0	Pada 2024-03-06, engine versi 1.3.1.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.3.1.0.	Maret 6, 2024

Memperbarui ke izin kebijakan AWS terkelola	Kebijakan `NeptuneReadOnlyAcess` dan `NeptuneFullAccess` terkelola sekarang menyertakan `Sid` (ID pernyataan) sebagai pengenal dalam pernyataan kebijakan.	Januari 22, 2024
Neptunus sekarang menawarkan penyimpanan I/O-Optimized	Dengan I/O-Optimized storage, you pay for the storage and instances you are using. The storage costs are higher than for standard storage, but you pay nothing for any I/O itu Anda gunakan.	13 Desember 2023
Perubahan Kebijakan Terkelola IAM untuk Neptunus	Kebijakan terkelola NeptuneConsoleFullAccessIAM telah diperbarui untuk memberikan izin yang diperlukan untuk berinteraksi dengan grafik Neptunus Analytics, NeptuneGraphReadOnlyAccesskebijakan baru telah ditambahkan untuk menyediakan akses hanya-baca ke sumber daya grafik Neptunus Analytics, dan kebijakan baru `AWSServiceRoleForNeptuneGraphPolicy` telah ditambahkan untuk memungkinkan grafik Neptunus Analytics mempublikasikan metrik dan log operasional dan penggunaan. CloudWatch	November 29, 2023

Versi mesin 1.3.0.0	Pada 2023-11-15, versi mesin 1.3.0.0 umumnya digunakan . Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.3.0.0.	15 November 2023
Neptunus diluncurkan di wilayah Israel (Tel Aviv)	Amazon Neptunus sekarang tersedia di wilayah Israel (Tel Aviv) (). `il-central-1`	13 November 2023
Posting blog tentang penerapan time-to-live dalam grafik properti Neptunus	Lihat Menerapkan Waktu untuk Hidup di Amazon Neptunus, Bagian 1: Grafik Properti oleh Melissa Kwok, Mike Havey, dan Kevin Phillips.	Oktober 27, 2023
Versi mesin 1.2.1.0.R7	Pada 2023-10-06, versi mesin 1.2.1.0.R7 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.1.0.R7.	Oktober 6, 2023

Versi mesin 1.2.0.0.R4	Pada 2023-09-29, versi mesin 1.2.0.0.R4 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.0.0.R4.	September 29, 2023
Versi mesin 1.2.0.1.R3	Pada 2023-09-27, versi mesin 1.2.0.1.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.0.1.R3.	27 September 2023
Versi mesin 1.2.1.0.R6	Pada 2023-09-12, versi mesin 1.2.1.0.R6 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.1.0.R6.	12 September 2023

Versi mesin 1.2.0.2.R6	Pada 2023-09-12, versi mesin 1.2.0.2.R6 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.0.2.R6.	12 September 2023
Posting blog tentang menggunakan strategi blue/green penyebaran untuk peningkatan mesin Neptunus	Lihat Meningkatkan ketersediaan Amazon Neptunus selama peningkatan mesin blue/green menggunakan penerapan oleh Ankit Gupta dan Abhishek Mishra.	11 September 2023
Versi mesin 1.2.1.0.R5	Pada 2023-09-02, versi mesin 1.2.1.0.R5 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.1.0.R5.	September 2, 2023

Versi mesin 1.2.0.2.R5	Pada 2023-08-16, versi mesin 1.2.0.2.R5 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.0.2.R5.	16 Agustus 2023
Versi mesin 1.2.1.0.R4	Pada 2023-08-10, versi mesin 1.2.1.0.R4 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.1.0.R4.	10 Agustus 2023
Posting blog tentang rilis mesin Neptunus 1.2.1.0	Lihat Menjelajahi rilis 1.2.1.0 yang dikemas fitur untuk Amazon Neptunus oleh Joy Wang, Kevin Phillips, Andrea Nassisi, dan Navtanay Sinha.	4 Agustus 2023
Posting blog tentang membangun solusi database multimodal dengan Neptunus	Lihat Merancang solusi database multimodel berbasis kasus penggunaan yang sangat skalabel menggunakan Amazon Neptunus oleh Mike Havey.	Juli 18, 2023

Posting blog tentang kasus penggunaan Neptunus Tanpa Server dan praktik terbaik	Lihat Kasus penggunaan dan praktik terbaik untuk mengoptimalkan biaya dan kinerja dengan Amazon Neptune Tanpa Server oleh Kevin Phillips dan Ankit Gupta.	28 Juni 2023
Versi mesin 1.2.1.0.R3	Pada 2023-06-13, versi mesin 1.2.1.0.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.1.0.R3.	13 Juni 2023
Posting blog tentang menghasilkan saran rekreasi secara real time	Lihat Menghasilkan saran untuk kegiatan rekreasi secara real time dengan Amazon Neptunus oleh Michael Meidlinger dan Nils Müller.	6 Juni 2023
Posting blog tentang pemodelan molekuler dengan Neptunus dan RDKit	Lihat Data Model molekuler SMILES dengan Amazon Neptunus dan oleh Graham RDKit Kutchek.	1 Juni 2023

Posting blog tentang menggunakan caching untuk mempercepat kinerja Neptunus (Bagian 3)	Lihat Mempercepat kinerja kueri grafik dengan caching di Amazon Neptunus, Bagian 3: Arsitektur caching seluruh cluster Neptunus dengan Amazon oleh Taylor Riggan, Abhishek Mishra, Melissa Kwok, dan Kelvin ElastiCache Lawrence.	26 Mei 2023
Posting blog tentang menggunakan caching untuk mempercepat kinerja Neptunus (Bagian 2)	Lihat Mempercepat kinerja kueri grafik dengan caching di Amazon Neptunus, Bagian 2: Fitur caching Neptunus tambahan oleh Taylor Riggan, Abhishek Mishra, Melissa Kwok, dan Kelvin Lawrence.	26 Mei 2023
Posting blog tentang menggunakan caching untuk mempercepat kinerja Neptunus (Bagian 1)	Lihat Mempercepat kinerja kueri grafik dengan caching di Amazon Neptunus, Bagian 1: Kueri dan buffer pool caching oleh Taylor Riggan, Abhishek Mishra, Melissa Kwok, dan Kelvin Lawrence.	26 Mei 2023
Posting blog tentang analisis rantai pasokan menggunakan Neptunus	Lihat Analisis dan visualisasi data rantai pasokan menggunakan Amazon Neptunus dan meja kerja Neptunus oleh Dhiraj Thakur dan Rajdip Chaudhur.	10 Mei 2023

Versi mesin 1.2.0.2.R4	Pada 2023-05-08, versi mesin 1.2.0.2.R4 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.0.2.R4.	8 Mei 2023
Neptunus diluncurkan di wilayah Timur Tengah (UEA)	Amazon Neptunus sekarang tersedia di wilayah Timur Tengah (UEA) ()`me-central-1` .	2 Mei 2023
Versi mesin 1.2.1.0.R2	Pada 2023-05-02, versi mesin 1.2.1.0.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.1.0.R2.	2 Mei 2023
Posting blog tentang membangun grafik pengetahuan di Neptunus dengan analisis video bertenaga AI menggunakan Media2Cloud	Lihat Membuat grafik pengetahuan di Amazon Neptunus dengan analisis video bertenaga AI menggunakan Media2Cloud oleh Mike Havey.	2 Mei 2023

Posting blog tentang bagaimana DevOcean membangun platform remediasi kerentanan menggunakan Neptunus	Lihat Cara DevOcean membangun platform manajemen remediasi kerentanan untuk aplikasi cloud-native menggunakan Amazon Neptunus oleh Gil Makmel dan Charles Ivie.	April 25, 2023
Posting blog tentang bagaimana Getir membangun sistem deteksi penipuan menggunakan Neptunus	Lihat Bagaimana Getir membangun sistem deteksi penipuan komprehensif menggunakan Amazon Neptunus dan Amazon DynamoDB oleh Berkay Berkman, Mahmut Turan, Mutlu Polatcan, Umut Cemal Kıraç, Yağız Yanıkoğlu, dan Esra Kayabali.	6 April 2023
Posting blog tentang bagaimana Wiz menata ulang keamanan cloud menggunakan Neptunus	Lihat Dunia adalah grafik: Bagaimana Wiz menata ulang keamanan cloud menggunakan grafik di Amazon Neptunus oleh Ami Luttwak dan Brad Bebee.	31 Maret 2023
Versi mesin 1.2.0.2.R3	Pada 2023-03-27, versi mesin 1.2.0.2.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.0.2.R3.	Maret 27, 2023

Posting blog tentang bagaimana Generasi CSC mendukung penemuan produk menggunakan Neptunus	Lihat Bagaimana Generasi CSC mendukung penemuan produk dengan grafik pengetahuan menggunakan Amazon Neptunus oleh Bobber Cheng, Ronit Rudra, dan Melissa Kwok.	21 Maret 2023
Versi mesin 1.2.1.0	Pada 2023-03-08, engine versi 1.2.1.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.2.1.0.	8 Maret 2023
Posting blog tentang menjelajahi fitur baru TinkerPop 3.6.x di Neptunus	Lihat Menjelajahi fitur baru Apache TinkerPop 3.6.x di Amazon Neptunus oleh Stephen Mallette.	8 Maret 2023
Posting blog tentang menggunakan penalaran semantik untuk menyimpulkan fakta baru dari grafik RDF Anda	Lihat Gunakan penalaran semantik untuk menyimpulkan fakta baru dari grafik RDF Anda dengan menginteg rasikan dengan Amazon RDFox Neptunus oleh Charles Ivie dan Diana Marks.	Februari 20, 2023
Posting blog tentang menganalisis data FHIR perawatan kesehatan dengan Neptunus	Lihat Menganalisis data FHIR perawatan kesehatan dengan Amazon Neptunus oleh Alena Schmickl.	13 Februari 2023

Posting blog tentang membangun solusi deteksi penipuan real-time menggunakan Neptunus ML	Lihat Membangun solusi deteksi penipuan real-time menggunakan Amazon Neptunus MLoleh Hua Shu dan Soji Adeshina.	Februari 8, 2023
Versi mesin 1.1.1.0.R7	Pada 2023-01-23, versi mesin 1.1.1.0.R7 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.1.1.0.R7.	23 Januari 2023
Graph-explorer dirilis	Graph-explorer adalah alat aplikasi web front-end open-source untuk memvisual isasikan data grafik. Lihat https://github.com/aws/penjelajah grafik.	Januari 3, 2023
Versi rilis pemeliharaan 1.1.0.0.R3	Pada 2022-12-23, rilis pemeliharaan versi engine 1.1.0.0.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.1.0.0.R3.	Desember 23, 2022

Meja kerja Neptunus sekarang beroperasi di Amazon Linux 2 dan 3. JupyterLab	Notebook grafik Neptunus sekarang berjalan di lingkungan Amazon Linux 2 dengan 3. JupyterLab Lihat Memigrasi buku catatan Neptunus Anda dari Jupyter JupyterLab ke 3 untuk informasi tentang cara pindah ke lingkungan baru ini.	21 Desember 2022
Posting blog tentang rekomendasi daya dan pencarian menggunakan grafik IMDb pengetahuan (bagian 3)	Lihat rekomendasi Daya dan cari menggunakan grafik IMDb pengetahuan — Bagian 3 oleh Divya Bhargavi, Soji Adeshina, Gaurav Rele, Karan Sindwani, Vidya Sagar Ravipati, dan Matthew Rhodes.	Desember 20, 2022
Posting blog tentang rekomendasi daya dan pencarian menggunakan grafik IMDb pengetahuan (bagian 2)	Lihat rekomendasi Daya dan cari menggunakan grafik IMDb pengetahuan — Bagian 2 oleh Matthew Rhodes, Soji Adeshina, Divya Bhargavi, Gaurav Rele, Karan Sindwani, dan Vidya Sagar Ravipati.	Desember 20, 2022
Posting blog tentang rekomendasi daya dan pencarian menggunakan grafik IMDb pengetahuan (bagian 1)	Lihat rekomendasi Daya dan cari menggunakan grafik IMDb pengetahuan — Bagian 1 oleh Gaurav Rele, Soji Adeshina, Divya Bhargavi, Karan Sindwani, Vidya Sagar Ravipati, dan Matthew Rhodes.	Desember 20, 2022

Neptunus Tanpa Server sekarang tersedia di wilayah baru AWS	Pada 2022-12-16, Neptune Serverless telah diluncurkan di wilayah baru berikut AWS : Kanada (Tengah), Eropa (Stockholm), Eropa (Frankfurt), Asia Pasifik (Singapura), dan Asia Pasifik (Sydney). Lihat kendala Amazon Neptunus Tanpa Server untuk semua wilayah tempat Neptunus Tanpa Server tersedia.	16 Desember 2022
Versi mesin 1.2.0.2.R2	Pada 2022-12-15, versi mesin 1.2.0.2.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.0.2.R2.	15 Desember 2022
Versi mesin 1.2.0.0.R3	Pada 2022-12-15, versi mesin 1.2.0.0.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.0.0.R3.	15 Desember 2022

Versi mesin 1.2.0.1.R2	Pada 2022-12-13, versi mesin 1.2.0.1.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.0.1.R2.	13 Desember 2022
Posting blog tentang merancang arsitektur analisis data besar pendidikan dengan AWS	Lihat Merancang arsitektur analisis data besar pendidikan dengan AWS oleh Lavanya Sood.	13 Desember 2022
Posting blog tentang bagaimana database grafik dapat meningkatkan pembelajaran	Lihat Bagaimana database grafik dapat meningkatkan pembelajaran oleh Lavanya Sood.	Desember 8, 2022
Posting blog tentang memuat data RDF ke Neptunus menggunakan Glue AWS	Lihat Memuat data RDF ke Amazon Neptunus dengan AWS Glue oleh Mike Havey dan Fabrizio Napolitano.	23 November 2022
Versi mesin 1.2.0.2	Pada 2022-11-20, engine versi 1.2.0.2 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.2.0.2.	November 20, 2022

Versi mesin 1.2.0.1	Pada 2022-10-26, engine versi 1.2.0.1 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.0.1.	26 Oktober 2022
Posting blog tentang deteksi penipuan menggunakan Neptunus	Lihat Memberdayakan deteksi penipuan di Delivery Hero with Amazon Neptunus oleh Wilson Tang, Amr Elnaggar, Matias Pons, Mohammad Azzam, Saurabh Deshpande, dan Luis Rodrigues Soares.	26 Oktober 2022
Posting blog tentang Neptunus Tanpa Server	Lihat Memperkenalkan Amazon Neptunus Tanpa Server — Database Grafik yang Dikelola Sepenuhnya yang Menyesuaikan Kapasitas Beban Kerja Anda oleh Danilo Poccia.	26 Oktober 2022
Posting blog tentang impor RDF berbasis peristiwa ke Neptunus menggunakan Lambda dan SPARQL UPDATE LOAD	Lihat Bagaimana NXP melakukan impor RDF berbasis peristiwa ke Amazon Neptunus menggunakan AWS Lambda dan SPARQL UPDATE LOAD oleh John Walker, Onno Buijs, Charles Ivie, dan Javy de Koning.	20 Oktober 2022

Versi mesin 1.2.0.0.R2	Pada 2022-10-14, versi mesin 1.2.0.0.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.2.0.0.R2.	14 Oktober 2022
Posting blog tentang pengkodean properti teks multi-bahasa di Neptunus	Lihat Menyandikan properti teks multi-bahasa di Amazon Neptunus untuk melatih model prediktif oleh Jiani Zhang.	14 Oktober 2022
Posting blog tentang pengujian otomatis akses data Neptunus	Lihat Pengujian otomatis akses data Amazon Neptunus dengan TinkerPop Apache Gremlin oleh Greg Biegel.	September 28, 2022
Versi mesin 1.1.1.0.R6	Pada 2022-09-23, versi mesin 1.1.1.0.R6 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.1.1.0.R6.	September 23, 2022

Posting blog tentang bagaimana Informatica® menggunakan Neptunus	Lihat Bagaimana Tata Kelola dan Katalog Data Cloud Informatica® menggunakan Amazon Neptunus untuk grafik pengetahuan oleh Tiju Titus John, Deepak Ram, dan Farooq Ashraf.	September 20, 2022
Posting blog tentang menggunakan Perspektif Neptunus dan Tom Sawyer untuk mengungkap penipuan keuangan	Temukan penipuan keuangan dengan Perspektif Amazon Neptunus dan Tom Sawyer oleh Janet M. Six, Manajer Produk Senior di Tom Sawyer Software.	30 Agustus 2022
Posting blog tentang membangun solusi deteksi penipuan real-time berbasis GNN menggunakan, SageMaker Neptunus, dan DGL	Lihat Membangun solusi deteksi penipuan real-time berbasis GNN menggunakan Amazon SageMaker, Amazon Neptunus, dan Deep Graph Library oleh Jian Zhang, Haozhu Wang, dan Mengxin Zhu.	Agustus 11, 2022
Posting blog tentang penggunaan tag sumber daya untuk menghentikan dan memulai sumber daya lingkungan Neptunus	Lihat Mengotomatiskan penghentian dan memulai sumber daya lingkungan Amazon Neptunus menggunakan tag sumber daya oleh Kevin Phillips.	1 Agustus 2022
Posting blog tentang artis yang berkontribusi pada Apache TinkerPop	Lihat Beyond Code: Artis yang Berkontribusi pada Apache TinkerPop oleh Stephen Mallette dan Ketrina Thompson.	1 Agustus 2022

Posting blog tentang kontrol akses berbutir halus untuk tindakan bidang data Neptunus	Lihat Kontrol Akses Berbutir Halus untuk tindakan bidang data Amazon Neptunus oleh Abhishek Mishra dan Ankit Gupta.	Juli 29, 2022
Posting blog tentang database global Neptunus	Lihat Memperkenalkan Basis Data Global Amazon Neptunus oleh Navtanay Sinha.	27 Juli 2022
Versi mesin 1.2.0.0	Pada 2022-07-21, engine versi 1.2.0.0 umumnya digunakan . Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkap nya tentang versi mesin ini, lihat Neptune Engine Release 1.2.0.0.	21 Juli 2022
Versi mesin 1.1.1.0.R5	Pada 2022-07-21, versi mesin 1.1.1.0.R5 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.1.1.0.R 5.	21 Juli 2022

Versi mesin 1.1.1.0.R4	Pada 2022-06-23, versi mesin 1.1.1.0.R4 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.1.1.0.R4.	Juni 23, 2022
Analisis grafik yang disederhanakan dan alur kerja pembelajaran mesin dengan integrasi Python	Sekarang Anda dapat menjalankan analisis grafik dan tugas pembelajaran mesin pada data grafik yang disimpan di Amazon Neptunus menggunakan integrasi Python sumber terbuka yang menyederhanakan ilmu data dan alur kerja ML. Lihat dokumentasi AWS Data Wrangler untuk Neptunus.	7 Juni 2022
Versi mesin 1.1.1.0.R3	Pada 2022-06-07, versi mesin 1.1.1.0.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.1.1.0.R3.	7 Juni 2022

Posting blog tentang mendeteksi berita palsu	Lihat Deteksi berita palsu media sosial menggunakan pembelajaran mesin grafik dengan Amazon Neptunus ML, oleh Hasan Shojaei dan Sarita Joshi.	19 Mei 2022
Posting blog tentang menggunakan SQL Server Integration Services (SSIS) dengan Neptunus	Lihat Temukan wawasan baru dari data Anda menggunakan SQL Server Integration Services (SSIS) dan Amazon Neptunus oleh Mesgana Gormley dan Melissa Kwok.	Mei 18, 2022
Versi rilis pemeliharaan 1.1.1.0.R2	Pada 2022-05-16, rilis pemeliharaan versi engine 1.1.1.0.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.1.1.0.R2.	Mei 16, 2022
Versi rilis pemeliharaan 1.1.0.0.R2	Pada 2022-05-16, rilis pemeliharaan versi engine 1.1.0.0.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.1.0.0.R2.	Mei 16, 2022

Versi rilis pemeliharaan 1.0.5.1.R4	Pada 2022-05-16, rilis pemeliharaan versi engine 1.0.5.1.R4 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.0.5.1.R4.	Mei 16, 2022
Versi rilis pemeliharaan 1.0.5.0.R5	Pada 2022-05-16, rilis pemeliharaan versi engine 1.0.5.0.R5 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.0.5.0.R5.	Mei 16, 2022
Posting blog tentang ketersediaan umum OpenCypher di Neptunus	Lihat Mengumumkan Ketersediaan Umum dukungan OpenCypher untuk Amazon Neptunus, oleh Navtanay Sinha dan Dave Bechberger.	22 April 2022

Versi mesin 1.1.1.0	Pada 2022-04-19, engine versi 1.1.1.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.1.1.0.	19 April 2022
Posting blog tentang garis keturunan data	Lihat Membangun garis keturunan data untuk data lake menggunakan AWS Glue, Amazon Neptunus, dan Spline, oleh Khoa Nguyen, Krithivasan Balasubramaniyan, dan Rahul Shaurya.	1 April 2022
Upgrade ke rilis mesin 1.1.0.0 diaktifkan kembali	Pada 2022-02-21, peningkatan ke rilis mesin 1.1.0.0 dinonaktifkan sementara. Mereka sekarang diaktifkan kembali.	22 Maret 2022
Posting blog tentang keandalan utilitas teknik	Lihat Menggunakan model sistem daya berbasis cloud, berdasarkan informasi data, untuk merekayasa keandalan utilitas, oleh Abhineet Parchure.	22 Maret 2022

Neptunus diluncurkan di Afrika (Cape Town)	Amazon Neptunus sekarang tersedia di Afrika (Cape Town) (). `af-south-1` Namun, dukungan notebook meja kerja Neptunus dinonaktifkan sementara pada konsol Neptunus di wilayah itu.	Februari 24, 2022
Posting blog tentang grafik berbasis model menggunakan OWL	Lihat Grafik berbasis model menggunakan OWL di Amazon Neptunus, oleh Mike Havey.	Februari 23, 2022
Posting blog tentang menjelajahi grafik pengetahuan semantik dengan Rhizomer	Lihat Jelajahi grafik pengetahuan semantik tanpa SPARQL menggunakan Amazon Neptunus dengan Rhizomer, oleh Roberto García.	Februari 22, 2022
Posting blog tentang grafik grid utilitas	Lihat Grafik jaringan utilitas aktif AWS, oleh Bobby Wilson dan Joseph Beer.	18 Februari 2022
Opsi pengkodean fitur teks Neptunus ML baru	Neptunus sekarang FastText mendukung dan pengkodean teks Sentence BERT untuk pelatihan. Lihat FastTextfitur dalam fitur Neptunus ML dan Sentence BERT di Neptunus ML.	Februari 15, 2022
Posting blog tentang kueri geospasial menggunakan OpenSearch Neptunus	Lihat Menggabungkan Amazon Neptunus dan Layanan OpenSearch Amazon untuk kueri geospasial, oleh Ross Gabay dan Abhilash Vinod.	1 Februari 2022

Posting blog tentang Penemuan Kejahatan Keuangan menggunakan Amazon EKS dan Neptunus	Lihat Penemuan Kejahatan Keuangan menggunakan Amazon EKS dan Database Grafik, oleh Severin Gassauer-Fleissner dan Zahi Ben Shabat.	1 Februari 2022
Volume cluster Neptunus sekarang dapat tumbuh menjadi 128 tebibytes (TiB)	Di semua wilayah yang didukung kecuali Tiongkok GovCloud dan, batas ukuran volume cluster Neptunus kini telah meningkat dari 64 TiB menjadi 128 Tib. Ini berlaku untuk semua rilis mesin yang dimulai dengan rilis 1.0.2.2. Lihat halaman penyimpanan Amazon Neptunus.	1 Februari 2022
Pencarian teks lengkap Neptunus sekarang terintegrasi dengan semua versi. OpenSearch	Lihat Pencarian teks lengkap di Amazon Neptunus menggunakan OpenSearch Layanan Amazon.	28 Januari 2022
Posting blog tentang menggunakan wadah Docker untuk menyebarkan buku catatan grafik	Lihat Menggunakan kontainer Docker untuk menyebarkan Notebook Grafik AWS, oleh Ganesh Sawhney dan Qiang Zhang.	Januari 22, 2022

Versi mesin 1.0.5.1.R3	Pada 2022-01-13, versi mesin 1.0.5.1.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.0.5.1.R3.	Januari 13, 2022
Posting blog tentang sistem rekomendasi berbasis grafik menggunakan Neptunus ML	Lihat Sistem rekomendasi berbasis grafik dengan Neptunus ML: Ilustrasi tentang tantangan prediksi tautan jejaring sosial, oleh Yanwei Cui dan Will Badr.	12 Januari 2022
Posting blog tentang autoscaling Neptunus	Lihat Skala otomatis database Amazon Neptunus Anda untuk memenuhi tuntutan beban kerja, oleh Navtanay Sinha dan Sudhanshu Gupta.	29 November 2021
Posting blog tentang analisis data grafik interaktif dan visualisasi	Lihat Membangun analisis dan visualisasi data grafik interaktif menggunakan Amazon Neptunus, Kueri Federasi Amazon Athena, dan Amazon, oleh Sandeep Veldi QuickSight dan Abhishek Mishra.	24 November 2021

Posting blog tentang mendeteksi penipuan identitas menggunakan pembelajaran mendalam berbasis grafik	Lihat Bagaimana Careem mendeteksi penipuan identitas menggunakan pembelajaran mendalam berbasis grafik dan Amazon Neptunus, oleh Kevin O'Brien, Kamran Habib, dan Will Badr.	23 November 2021
Versi mesin 1.1.0.0	Pada 2021-11-19, versi mesin 1.1.0.0 umumnya digunakan . Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.1.0.0.	November 19, 2021
Posting blog tentang memusatkan perlindungan dan kepatuhan data	Lihat Memusatkan perlindungan dan kepatuhan data di Amazon Neptunus dengan Cadangan, oleh AWS Brian O'Keefe.	November 8, 2021
Posting blog tentang memerangi penipuan dan pembayaran yang tidak tepat	Lihat Memerangi penipuan dan pembayaran yang tidak tepat secara real-time pada skala pengeluaran federal oleh Vladi Royzman dan Spencer Smith.	2 November 2021
Posting blog tentang mencegah pendaftaran akun palsu	Lihat Mencegah pendaftaran akun palsu secara real time dengan AI menggunakan Amazon Fraud Detector, oleh Anjan Biswas.	29 Oktober 2021

Versi mesin 1.0.5.1.R2	Pada 2021-10-26, versi mesin 1.0.5.1.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.0.5.1.R2.	26 Oktober 2021
Posting blog tentang HawkEye 360 memprediksi risiko kapal menggunakan Deep Graph Library	Lihat HawkEye 360 memprediksi risiko kapal menggunakan Deep Graph Library dan Amazon Neptunus oleh Tim Pavlick, Ian Avilez, Dan Ford, dan Gaurav Rele.	Oktober 15, 2021
Versi mesin 1.0.5.1	Pada 2021-10-01, versi mesin 1.0.5.1 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.0.5.1.	1 Oktober 2021
Posting blog tentang mengapa pengembang suka TinkerPop	Lihat Mengapa pengembang seperti Apache TinkerPop, kerangka kerja open source untuk komputasi grafik, oleh Brad Bebee, Kelvin Lawrence, dan Stephen Mallette.	27 September 2021

Versi mesin 1.0.5.0.R3	Pada 2021-09-15, versi mesin 1.0.5.0.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.0.5.0.R3.	15 September 2021
Posting blog tentang membangun solusi penemuan data dengan Amundsen dan Neptunus	Lihat Membangun solusi penemuan data dengan Amundsen dan Amazon Neptunus, oleh Peter Hanssens dan Don Simpson.	8 September 2021
Neptunus telah memperbarui stream-poller untuk mendukung permintaan pencarian teks lengkap non-string	Termasuk dalam rilis ini banyak perbaikan untuk pencarian teks lengkap, termasuk dukungan untuk pengindeksan nilai properti yang bukan string. Lihat OpenSearch Pengindeksan non-string di Amazon Neptunus.	23 Agustus 2021
Versi mesin 1.0.5.0.R2	Pada 2021-08-16, versi mesin 1.0.5.0.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.0.5.0.R2.	16 Agustus 2021

Versi mesin 1.0.4.2.R5	Pada 2021-08-16, versi mesin 1.0.4.2.R5 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptunus 1.0.4.2.R5.	16 Agustus 2021
Posting blog tentang dukungan Protokol Graph Store di Neptunus	Lihat Memperkenalkan dukungan Protokol Toko Grafik untuk Amazon Neptunus, oleh Chris Smith.	2 Agustus 2021
Posting blog tentang fitur baru di Neptunus ML	Lihat Temukan lebih banyak wawasan dalam grafik Anda dengan fitur baru dari Amazon Neptunus ML, oleh Soji Adeshina.	30 Juli 2021
Posting blog tentang mendapatkan prediksi lebih cepat dengan Neptunus ML	Lihat Dapatkan prediksi untuk mengembangkan data grafik lebih cepat dengan Amazon Neptunus ML, oleh Soji Adeshina.	30 Juli 2021
Posting blog tentang pembelajaran mesin grafik yang lebih mudah dan lebih cepat dengan Neptunus ML	Lihat Pembelajaran mesin grafik yang lebih mudah dan lebih cepat dengan Amazon Neptunus ML, oleh Soji Adeshina.	30 Juli 2021

Posting blog tentang dukungan Neptune OpenCypher	Lihat Mengumumkan OpenCypher untuk Amazon Neptunus: Membangun aplikasi grafik yang lebih baik dengan OpenCypher dan Gremlin bersama-sama, oleh Brad Bebee.	29 Juli 2021
Versi mesin 1.0.5.0	Pada 2021-07-27, engine versi 1.0.5.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Neptune Engine Release 1.0.5.0.	27 Juli 2021
Versi mesin 1.0.4.2.R4	Per 2021-07-23, versi mesin 1.0.4.2.R4 umumnya sedang di-deploy. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.4.2.R4.	23 Juli 2021
Neptunus diluncurkan di Tiongkok (Beijing)	Amazon Neptune sekarang tersedia di China (Beijing) (`cn-north-1`).	21 Juli 2021

Versi mesin 1.0.4.2.R3	Per 2021-06-28, versi mesin 1.0.4.2.R3 secara sedang di-deploy. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.4.2.R3.	28 Juni 2021
Posting blog tentang bagaimana Dream11 meningkatkan jaringan sosial mereka menggunakan Neptunus	Lihat Pelajari bagaimana Dream11, platform olahraga fantasi terbesar di dunia, meningkatkan skala jejaring sosial mereka dengan Amazon Neptunus dan Amazon. ElastiCache	25 Juni 2021
Posting blog tentang mengubah data menjadi pengetahuan dengan PoolParty Neptunus dan Semantic Suite	Lihat Mengubah data menjadi pengetahuan dengan PoolParty Semantic Suite dan Amazon Neptunus, oleh Ioanna Lytra dan Albin Ahmeti.	16 Juni 2021
Posting blog tentang menggunakan Neptunus untuk menjelajahi basis pengetahuan UniProt	Lihat Menjelajahi basis pengetahuan UniProt protein dengan AWS Open Data dan Amazon Neptunus, oleh Eric Greene, Rafa Xu, dan Yuan Shi.	10 Juni 2021

Posting blog tentang menggunakan Neptunus untuk analisis risiko berbasis data	Lihat Catatan Lapangan: Analisis Risiko Berbasis Data dengan Amazon Neptunus dan Layanan OpenSearch Amazon, oleh Adriaan de Jonge dan Rohit Satyanara yana.	10 Juni 2021
Versi mesin 1.0.4.2.R2	Per 2021-06-01, versi mesin 1.0.4.2.R2 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.4.2.R2.	1 Juni 2021
Posting blog tentang menggunakan Neptunus untuk memvisualisasikan infrastruktur Anda AWS	Lihat Visualisasikan AWS Infrastruktur Anda dengan Amazon Neptunus dan Config, oleh AWS Rohan Raizada dan Amey Dhavle.	25 Mei 2021
Posting blog tentang konfigurasi untuk menggunakan Lensa Data dengan Neptunus	Lihat Mengonfigurasi AWS layanan untuk membuat grafik pengetahuan di Amazon Neptunus menggunakan Lensa Data, oleh Russell Waterson.	5 Mei 2021
Posting blog tentang membangun grafik pengetahuan di Neptunus menggunakan Lensa Data	Lihat Membangun grafik pengetahuan di Amazon Neptune menggunakan Data Lens, oleh Russell Waterson.	5 Mei 2021

Versi mesin 1.0.1.0, 1.0.1.1, dan 1.0.1.2 sekarang tidak digunakan lagi	Mulai sekarang, tidak ada instans DB baru akan dibuat menggunakan salah satu versi mesin ini, atau patch yang terkait dengan mereka.	26 April 2021
Terjemahan bahasa Inggris dari studi kasus tentang Kementerian Ekonomi, Perdagangan, dan Industri Jepang menggunakan Neptunus	Lihat Kementerian Ekonomi, Perdagangan, dan Industri Jepang Menguatkan Database Pencarian Informasi Perusahaan GBizInfo dengan AWS.	31 Maret 2021
Posting blog tentang menggunakan Neptunus dengan Amazon Comprehend dan Lex	Lihat Supercharge grafik pengetahuan Anda menggunakan Amazon Neptune, Amazon Comprehend, dan Amazon Lex, oleh Dave Bechberger.	31 Maret 2021
Posting blog tentang menggunakan fungsi Lambda dengan Neptunus	Lihat Menggunakan fungsi AWS Lambda dengan Amazon Neptunus, oleh Ian Robinson.	26 Maret 2021
Versi mesin 1.0.4.1.R1.1	Seperti 2021-03-22, versi mesin 1.0.4.1.R1.1 umumnya di-deploy. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.4.1.R1.1.	22 Maret 2021

Versi mesin 1.0.4.1.R2.1	Seperti 2021-03-11, versi mesin 1.0.4.1.R2.1 umumnya di-deploy. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.4.1.R2.1.	11 Maret 2021
Posting blog tentang penggunaan notebook grafik open source Neptunus untuk visualisasi grafik	Lihat Memulai dengan buku catatan grafik sumber terbuka untuk visualisasi grafik, oleh Joy Wang, Ora Lassila, dan Stephen Mallette.	10 Maret 2021
Tutorial tentang menginteg rasikan Neptunus dengan penemuan data Amundsen dan mesin metadata	Lihat Cara menggunakan Amundsen dengan Amazon Neptune, oleh Andrew Ciambrone.	2 Maret 2021
Versi mesin 1.0.4.1.R2	Seperti 2021-02-24, versi mesin 1.0.4.1.R2 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.4.1.R2.	24 Februari 2021

Versi mesin 1.0.4.0.R2	Seperti 2021-02-24, versi mesin 1.0.4.0.R2 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.4.0.R2.	24 Februari 2021
Versi mesin 1.0.3.0.R3	Per 2021-02-19, versi mesin 1.0.3.0.R3 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.3.0.R3.	19 Februari 2021
Versi mesin 1.0.2.2.R6	Per 2021-02-19, versi mesin 1.0.2.2.R6 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.2.2.R6.	19 Februari 2021

Posting blog tentang membuat grafik pengetahuan menggunakan Amazon Comprehend peristiwa	Lihat Membangun grafik pengetahuan di Amazon Neptune menggunakan peristiwa Amazon Comprehend, oleh Brian O'Keefe, Graham Horwood, dan Navtanay Sinha.	19 Januari 2021
Posting blog tentang mengaktifkan aplikasi data grafik kode rendah	Lihat Mengaktifkan aplikasi data grafik kode rendah dengan Amazon Neptune dan Graphistry, oleh Leo Meyerovich, Dave Bechberger, dan Taylor Riggan.	18 Januari 2021
Menambahkan dokumentasi notebook untuk memulai dengan data grafik.	Menambahkan bagian yang terintegrasi dengan workbench Neptune yang membantu Anda memulai membuat data grafik dan mengembangkan aplikasi grafik tanpa harus memutar klaster Neptune sampai Anda siap.	15 Januari 2021
Posting blog tentang mengatur ulang data grafik Neptunus Anda dalam hitungan detik	Lihat Mengatur ulang data grafik Anda di Amazon Neptune dalam hitungan detik, oleh Niraj Jetly dan Navtanay Sinha.	17 Desember 2020

Posting blog tentang bagaimana Novartis AG menggunakan SageMaker dan Neptunus dengan BERT	Lihat Novartis AG menggunakan Amazon SageMaker dan Amazon Neptunus untuk membangun dan memperkaya grafik pengetahuan menggunakan BERT, oleh Othmane Hamzaoui, Fatema Alkhanaizi, dan Viktor Malesevic.	14 Desember 2020
Posting blog tentang membangun grafik pengetahuan dengan jaringan topik	Lihat Membangun grafik pengetahuan dengan jaringan topik di Amazon Neptune, oleh Edward Brown, Kepala Proyek AI, Eduardo Piairo, Arsitek, Marcia Oliveira, Lead Data Scientist, dan Jack Hampson, CEO di Deeper Insights.	14 Desember 2020
Versi mesin 1.0.4.1	Per 2020-12-08, versi mesin 1.0.4.1 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.4.1.	8 Desember 2020
Posting blog tentang memulai dengan Neptunus ML	Lihat Cara memulai dengan Neptune ML, oleh George Karypis, Dave Bechberger, dan Karthik Bharathy.	8 Desember 2020

Neptunus sekarang memiliki API reset cepat	Menggunakan API reset cepat, Anda dapat dengan cepat dan mudah menghapus semua data dalam klaster DB. Lihat API reset cepat.	4 Desember 2020
Posting blog tentang membangun grafik pengetahuan biologis di Pendulum	Lihat Membangun grafik pengetahuan biologis di Pendulum menggunakan Amazon Neptune, oleh Connor Skennerton.	26 November 2020
Posting blog tentang fitur baru TinkerPop 3.4.8 di Neptunus	Lihat Menjelajahi fitur baru Apache TinkerPop 3.4.8 di Amazon Neptunus, oleh Stephen Mallette.	18 November 2020
Posting blog tentang menggunakan layanan pencarian Amazon Kendra dengan Neptunus	Lihat Memasukkan grafik pengetahuan perusahaan Anda ke Amazon Kendra, oleh Yazdan Shirvany, Mohit Mehta, dan Dipto Chakravarty.	17 November 2020
Pemberitahuan acara sekarang tersedia	Neptune sekarang mendukung pemberitahuan peristiwa yang dapat Anda gunakan untuk lebih mudah memantau klaster DB. Lihat Menggunakan Notifikasi Peristiwa Neptune, oleh.	29 Oktober 2020
Titik akhir bea cukai sekarang tersedia	Neptune sekarang mendukung titik akhir kustom untuk kontrol yang lebih besar saat menghubungkan ke instans DB. Lihat Menghubungkan ke Titik Akhir Amazon Neptune.	29 Oktober 2020

Posting blog tentang menggunakan AWS Database Migration Service (DMS) untuk mengisi grafik Neptunus Anda	Lihat Mengisi grafik Anda di Amazon Neptune dari database relasional menggunakan Database AWS Migration Service (DMS) — Bagian 4: Menyatukan semuanya, oleh Chris Smith.	22 Oktober 2020
Posting blog tentang menggunakan AWS Database Migration Service (DMS) untuk mengisi grafik Neptunus Anda	Lihat Mengisi grafik Anda di Amazon Neptunus dari database relasional menggunakan Database AWS Migration Service (DMS) — Bagian 3: Merancang Model RDF, oleh Chris Smith.	22 Oktober 2020
Posting blog tentang menggunakan AWS Database Migration Service (DMS) untuk mengisi grafik Neptunus Anda	Lihat Mengisi grafik Anda di Amazon Neptunus dari database relasional menggunakan Database AWS Migration Service (DMS) — Bagian 2: Merancang model grafik properti, oleh Chris Smith.	22 Oktober 2020
Posting blog tentang menggunakan AWS Database Migration Service (DMS) untuk mengisi grafik Neptunus Anda	Lihat Mengisi grafik Anda di Amazon Neptunus dari database relasional menggunakan Database AWS Migration Service (DMS) — Bagian 1: Mengatur panggung, oleh Chris Smith.	22 Oktober 2020

Versi mesin 1.0.4.0	Per 2020-10-12, versi mesin 1.0.4.0 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.4.0.	12 Oktober 2020
Versi mesin 1.0.3.0.R2	Per 2020-10-12, versi mesin 1.0.3.0.R2 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.3.0.R2.	12 Oktober 2020
Versi mesin 1.0.2.2.R5	Per 2020-10-12, versi mesin 1.0.2.2.R5 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.2.2.R5.	12 Oktober 2020
Posting blog tentang mengonfigurasi VPC Anda untuk kueri federasi SPARQL	Lihat Konfigurasi Amazon VPC untuk Kueri Gabungan SPARQL 1.1 dengan Amazon Neptune, oleh Charles Ivie.	12 Oktober 2020

Posting blog tentang menulis penghapusan cascading SPARQL	Lihat Menulis cascading delete di SPARQL, oleh Ora Lassila.	5 Oktober 2020
Posting blog tentang AWS sumber grafik menggunakan Neptunus	Lihat Grafik AWS sumber daya Anda dengan Amazon Neptunus, oleh Dave Bechberger.	Senin, 28 September 2020
Posting blog tentang membangun pemetaan terminologi MedDRA untuk farmakovigilans dan pelaporan efek samping menggunakan Neptunus	Lihat Membangun Amazon Neptune berdasarkan pemetaan terminologi MedDRA untuk farmacovigilance dan pelaporan kejadian buruk, oleh Vaijayanti Joshi, Deven Atnoor, Ph.D, dan Sudhanshu Malhotra.	24 September 2020
Posting blog tentang membangun grafik pengetahuan dari gudang data menggunakan Neptunus, untuk melengkapi intelijen komersial	Lihat Melengkapi Intelijen Komersial dengan Membangun Grafik Pengetahuan dari Gudang Data dengan Amazon Neptune, oleh Shahria Hossain dan Mikael Graindorge.	23 September 2020
Posting blog tentang load balancing menggunakan klien Neptunus Gremlin	Lihat penyeimbangan beban kueri grafik menggunakan Klien Amazon Neptune Gremlin, oleh Ian Robinson.	16 September 2020
Posting blog tentang personalisasi digital menggunakan grafik identitas di Cox Automotive	Lihat Personalisasi digital skala Cox Otomotif menggunakan grafik identitas yang didukung oleh Amazon Neptune, oleh Carlos Rendon dan Niraj Jetly.	16 September 2020

Posting blog tentang penyaringan kolaboratif pada data Yelp	Lihat Menggunakan penyaringan kolaboratif pada data Yelp untuk membangun sistem rekomendasi di Amazon Neptune, oleh Chad Tindel.	8 September 2020
Posting blog tentang visualisasi kueri-hasil di Amazon Neptunus	Lihat Visualisasikan hasil kueri menggunakan workbench Amazon Neptune, oleh Kelvin Lawrence.	2 September 2020
Neptunus merilis visualisasi grafik	Amazon Neptune sekarang menyediakan kemampuan visualisasi grafik yang luas di notebook Jupyter di workbench Neptune, bersama dengan sejumlah fitur baru yang membuat notebook lebih mudah digunakan. Lihat Visualisasi grafik.	12 Agustus 2020
Neptunus diluncurkan di Amerika Selatan (São Paulo)	Amazon Neptune sekarang tersedia di Amerika Selatan (São Paulo) (`sa-east-1`).	6 Agustus 2020
Neptunus diluncurkan di Asia Pasifik (Hong Kong)	Amazon Neptune kini tersedia di Asia Pacific (Hong Kong) (`ap-east-1`).	6 Agustus 2020

Versi mesin 1.0.3.0	Per 2020-08-03, versi mesin 1.0.3.0 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.3.0.	3 Agustus 2020
Versi mesin 1.0.2.2.R4	Per 2020-07-23, versi mesin 1.0.2.2.R4 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.2.2.R4.	23 Juli 2020
Posting blog tentang penelusuran kontak otomatis Zerobase menggunakan Amazon Neptunus	Lihat Zerobase menciptakan pelacakan kontak pribadi, aman, dan otomatis menggunakan Amazon Neptune, oleh David Harris dan Aron Szanto.	13 Juli 2020
Neptunus diluncurkan di AS Barat (California Utara)	Amazon Neptune sekarang tersedia di US West (N. California) (`us-west-1`).	9 Juli 2020

Amazon Neptunus mendukung kontrol akses berbasis tag	Anda sekarang dapat menggunakan AWS tag dalam kebijakan IAM untuk mengontrol akses ke database Neptunus Anda. Lihat Kontrol akses berbasis tanda di Amazon Neptune.	7 Juli 2020
Poller aliran Java sekarang tersedia	Amazon Neptune sekarang mendukung versi Java dari poller aliran lambda untuk aliran Neptune serta Python one. Lihat Tambahkan detail tentang tumpukan konsumen Neptune Streams yang Anda buat.	6 Juli 2020
Posting blog tentang grafik pengetahuan AWS COVID-19	Lihat Membangun dan menanyakan grafik pengetahuan AWS COVID-19, oleh Ninad Kulkarni, Colby Wise, George Price, dan Miguel Romero.	1 Juli 2020
Versi mesin 1.0.1.1	Per 2020-06-26, versi mesin 1.0.1.1 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.1.1.	26 Juni 2020

Posting blog tentang migrasi dari Blazegraph ke Amazon Neptunus	Lihat Pindah ke cloud: Migrasi Blazegraph ke Amazon Neptune, oleh Dave Bechberger.	25 Juni 2020
Posting blog tentang mengubah pengambilan data dari Neo4j ke Amazon Neptunus	Lihat Perubahan perekaman data dari Neo4j ke Amazon Neptune menggunakan Amazon Managed Streaming for Apache Kafka, oleh Sanjeet Sahay.	22 Juni 2020
Posting blog tentang bagaimana Waves menggunakan Amazon Neptunus	Lihat Bagaimana Waves menjalankan kueri pengguna dan rekomendasi dalam skala dengan Amazon Neptune, oleh Pavel Vasilyev.	16 Juni 2020
Versi mesin 1.0.1.2	Per 2020-06-10, versi mesin 1.0.1.2 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.1.2.	10 Juni 2020
Posting blog tentang membangun repositori pengetahuan pelanggan	Lihat Membangun repositori pengetahuan 360 pelanggan dengan Amazon Neptune dan Amazon Redshift, oleh Ram Bhandarkar.	9 Juni 2020

Posting blog tentang bagaimana Gunosy menggunakan Amazon Neptunus	Lihat Bagaimana Gunosy membangun fitur komentar di News Pass menggunakan Amazon Neptune oleh Yosuke Uchiyama.	8 Juni 2020
Posting blog tentang grafik pengetahuan AWS COVID-19	Lihat Membangun dan menanyakan grafik pengetahuan AWS COVID-19 oleh Ninad Kulkarni, Colby Wise, George Price, dan Miguel Romero.	2 Juni 2020
Posting blog tentang menjelajahi penelitian COVID-19 menggunakan Amazon Neptunus	Lihat Mengeksplorasi penelitian ilmiah tentang COVID-19 dengan Amazon Neptune, Amazon Comprehend Medical, dan Browser Database Grafik Tom Sawyer oleh George Price, Colby Wise, Miguel Romero, dan Ninad Kulkarni.	2 Juni 2020
Anda sekarang dapat memuat data ke Neptunus menggunakan AWS DMS	Lihat Menggunakan Layanan Migrasi AWS Database untuk memuat data ke Amazon Neptune dari penyimpanan data yang berbeda.	1 Juni 2020
Engine versi 1.0.2.0 tidak digunakan lagi	Mesin Amazon Neptune versi 1.0.2.0 sekarang sudah usang. Klaster yang berjalan pada versi mesin ini akan ditingkatkan ke versi 1.0.2.1 secara otomatis selama jendela pemeliharaan pertama setelah 1 Juni 2020.	19 Mei 2020

Posting blog tentang membangun grafik identitas pelanggan menggunakan Neptunus	Lihat Membangun grafik identitas pelanggan dengan Amazon Neptune oleh Rajesh Wunnava dan Taylor Riggan.	12 Mei 2020
Versi mesin 1.0.2.0.R3	Per 2020-05-05, versi mesin 1.0.2.0.R3 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.2.0.R3.	5 Mei 2020
Versi mesin 1.0.2.1.R6	Per 2020-04-22, versi mesin 1.0.2.1.R6 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.2.1.R6.	22 April 2020
Posting blog tentang migrasi data dari Neo4j ke Neptunus	Lihat Memigrasi database grafik Neo4j ke Amazon Neptune dengan utilitas yang sepenuhnya otomatis oleh Sanjeet Sahay.	13 April 2020

Posting blog tentang menurunkan biaya pembuatan aplikasi grafik dengan Neptunus	Lihat Menurunkan biaya pembuatan aplikasi grafik hingga 76% dengan instans Amazon Neptune T3 oleh Karthik Bharathy dan Brad Bebee.	9 April 2020
Neptunus menawarkan kelas instans T3 burstable	Anda sekarang dapat membuat instans burstable Amazon Neptune T3 untuk tujuan pengembangan dan pengujian yang hemat biaya. Lihat Kelas Instans Burstable Neptune T3.	8 April 2020
Versi mesin 1.0.2.2.R2	Per 2020-04-02, versi mesin 1.0.2.2.R2 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.2.2.R2.	2 April 2020
Posting blog tentang grafik ketergantungan investasi di EDGAR	Lihat Membuat grafik ketergantungan investasi dengan Amazon Neptune oleh Lawrence Verdi.	17 Maret 2020
Neptunus diluncurkan di Eropa (Paris)	Amazon Neptune sekarang tersedia di Eropa (Paris) (eu-west-3).	11 Maret 2020

Versi mesin 1.0.2.2	Per 2020-03-09, versi mesin 1.0.2.2 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.2.2.	9 Maret 2020
Menghentikan dan Memulai Ulang Cluser DB	Anda sekarang dapat menghentikan klaster DB selama 7 hari menggunak an konsol Neptune, dan kemudian melakukan restart ketika Anda membutuhk annya lagi. Saat klaster Anda dihentikan, Anda dikenakan biaya untuk penyimpanan klaster, snapshot manual, dan penyimpanan cadangan otomatis dalam jendela penyimpanan yang ditentuka n, tapi tidak untuk jam instans DB. Lihat Menghentikan dan Merestrat Klaster DB Amazon Neptune.	19 Februari 2020

Video tentang grafik sosial di Nike	Dengarkan saat Todd Escalona AWS berbicara dengan Marc Wangenheim, Manajer Teknik Senior di Nike, tentang bagaimana perusahaan memberdayakan sejumlah aplikasi melalui grafik sosial yang dibangun di Amazon Neptunus. Lihat Nike: Grafik Sosial Pada Skala dengan Amazon Neptune.	11 Februari 2020
Cluster Neptunus sekarang dapat dikonfigurasi untuk memerlukan koneksi SSL	Di wilayah yang masih mendukung koneksi HTTP, SSL sekarang diaktifkan secara default di semua grup parameter baru. Tidak ada perubahan pada grup parameter yang ada, tetapi Anda dapat memaksa klien untuk menggunakan SSL dengan mengubah parameter `neptune_enforce_ssl` ke 1. Lihat Enkripsi dalam Transit: Menghubungkan ke Neptune Menggunakan SSL/HTTPS untuk informasi tentang cara mengaktifkan koneksi HTTP untuk sebuah klaster di wilayah yang masih mendukungnya. Lihat Parameter yang dapat Anda gunakan untuk mengonfigurasi Amazon Neptune untuk deskripsi parameter klaster dan instans.	10 Februari 2020

Anda sekarang dapat menentukan versi mesin dan perlindungan penghapusan di template Neptunus CloudFormation	Amazon Neptunus telah memperbarui CloudFormation templatnya untuk menyertakan parameter `AWS::Neptune::DBCluster.EngineVersion` yang memungkinkan Anda menentukan versi engine tertentu untuk cluster DB baru Anda, dan parameter `AWS::Neptune::DBCluster.DeletionProtection` yang memungkinkan Anda mengaktifkan perlindungan penghapusan untuk itu.	9 Februari 2020
Perlindungan penghapusan	Amazon Neptune telah memberikan perlindungan penghapusan untuk klaster dan instans DB. Selama perlindungan penghapusan diaktifkan pada klaster atau instans DB, Anda tidak dapat menghapusnya. Lihat Anda tidak dapat menghapus instans DB saat Perlindungan Penghapusan diaktifkan.	20 Januari 2020
Neptunus diluncurkan di Tiongkok (Ningxia)	Amazon Neptune sekarang tersedia di China (Ningxia) (cn-northwest-1).	15 Januari 2020
Versi mesin 1.0.2.1.R4	Patch R4 untuk mesin versi 1.0.2.1 umumnya tersedia. Untuk informasi selengkapnya, lihat Rilis Mesin Neptune 1.0.2.1.R4.	20 Desember 2019

Versi mesin 1.0.2.1.R3	Patch R3 untuk mesin versi 1.0.2.1 umumnya tersedia. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.2.1.R3.	12 Desember 2019
Posting blog tentang menggunakan Neptunus untuk menganalisis umpan media sosial	Lihat Menganalisis umpan media sosial menggunakan Amazon Neptune.	27 November 2019
Versi mesin 1.0.2.1.R2	Patch R2 untuk mesin versi 1.0.2.1 umumnya tersedia. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.2.1.R2.	25 November 2019
Versi mesin 1.0.2.1.R1	Mesin Amazon Neptune versi 1.0.2.1.R1 umumnya tersedia. Untuk informasi selengkapnya, lihat Rilis Mesin Neptune 1.0.2.1.	22 November 2019
Versi mesin 1.0.2.0.R2	Patch R2 untuk mesin versi 1.0.2.0 umumnya tersedia. Untuk informasi selengkapnya tentang versi mesin ini, lihat Rilis Mesin Neptune 1.0.2.0.R2.	21 November 2019
Posting blog tentang sesi dan lokakarya Neptunus di re:Invent 2019	Lihat panduan Anda untuk sesi Amazon Neptunus, lokakarya, dan pembicaraan kapur di re:Invent 2019. AWS	20 November 2019

Versi mesin 1.0.2.0.R1	Mesin Amazon Neptune versi 1.0.2.0.R1 umumnya tersedia. Untuk informasi selengkap nya, lihat Rilis Mesin Neptune 1.0.2.0.	8 November 2019
Posting blog tentang menangkap perubahan grafik menggunakan Neptune Streams	Lihat Menangkap Perubahan Grafik menggunakan Neptune Streams.	6 November 2019
Versi mesin 1.0.1.0.200502.0	Mesin Amazon Neptune versi 1.0.1.0.200502.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200502.0.	31 Oktober 2019
Neptunus diluncurkan di Timur Tengah (Bahrain)	Amazon Neptune sekarang tersedia di Middle East (Bahrain) (me-south-1).	30 Oktober 2019
Neptunus diluncurkan di Kanada (Tengah)	Amazon Neptune sekarang tersedia di Canada (Central) (ca-central-1).	30 Oktober 2019
Posting blog tentang fitur SPARQL Streams baru Neptunus dan dukungan kueri federasi SPARQL	Lihat Amazon Neptune merilis Streams, kueri gabungan SPARQL untuk grafik dan banyak lagi.	17 Oktober 2019
Versi mesin 1.0.1.0.200463.0	Mesin Amazon Neptune versi 1.0.1.0.200463.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200463.0.	15 Oktober 2019

Versi mesin 1.0.1.0.200457.0	Mesin Amazon Neptune versi 1.0.1.0.200457.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200457.0.	19 September 2019
Posting blog tentang fitur penjelasan SPARQL baru Neptunus	Lihat Menggunakan explain SPARQL untuk memahami eksekusi kueri di Amazon Neptune.	17 September 2019
Posting blog tentang dukungan Neptunus untuk 3.4 TinkerPop	Lihat Amazon Neptunus sekarang TinkerPop mendukung fitur 3.4.	6 September 2019
Posting blog tentang menggunakan PyTorch Neptunus dengan di Amazon SageMaker	Lihat Pengalaman 'shop-by-style' yang dipersonalisasi menggunakan PyTorch di Amazon SageMaker dan Amazon Neptunus.	22 Agustus 2019
Posting blog tentang menggunakan Neptunus dengan AWS AppSync dan Amazon Elasticache	Lihat Mengintegrasikan sumber data alternatif dengan AWS AppSync: Amazon Neptunus dan Amazon. ElastiCache	22 Agustus 2019
Neptunus diluncurkan AWS GovCloud di (AS-Timur)	Amazon Neptunus sekarang tersedia AWS GovCloud di (AS-Timur) (-1). us-gov-east	21 Agustus 2019
Neptunus diluncurkan AWS GovCloud di (AS-Barat)	Amazon Neptunus sekarang tersedia AWS GovCloud di (AS-Barat) (-1). us-gov-west	14 Agustus 2019

Versi mesin 1.0.1.0.200369.0	Mesin Amazon Neptune versi 1.0.1.0.200369.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200369.0.	13 Agustus 2019
Versi mesin 1.0.1.0.200366.0	Mesin Amazon Neptune versi 1.0.1.0.200366.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200366.0.	26 Juli 2019
Posting blog tentang menggunakan PyTorch Neptunus dengan di Amazon SageMaker	Lihat Pengalaman 'shop-by-style' yang dipersonalisasi menggunakan PyTorch di Amazon SageMaker dan Amazon Neptunus.	3 Juli 2019
Versi mesin 1.0.1.0.200348.0	Mesin Amazon Neptune versi 1.0.1.0.200348.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200348.0.	2 Juli 2019
Neptunus diluncurkan di Eropa (Stockholm)	Amazon Neptune sekarang tersedia di Eropa (Stockholm) (eu-north-1).	27 Juni 2019
Neptunus sekarang dapat mempublikasikan log audit ke Log CloudWatch	Untuk informasi selengkapnya, lihat Menerbitkan Log Neptunus ke Log Amazon. CloudWatch	Selasa, 18 Juni 2019
Versi mesin 1.0.1.0.200310.0	Mesin Amazon Neptune versi 1.0.1.0.200310.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200310.0.	12 Juni 2019

Posting blog LifeOmic tentang JupiterOne	Lihat LifeOmicHow's JupiterOne menyederhanakan operasi keamanan dan kepatuhan dengan Amazon Neptunus.	2 Mei 2019
Neptunus diluncurkan di Asia Pasifik (Seoul)	Amazon Neptune sekarang tersedia di Asia Pacific (Seoul) (ap-northeast-2).	1 Mei 2019
Versi mesin 1.0.1.0.200296.0	Mesin Amazon Neptune versi 1.0.1.0.200296.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200296.0.	1 Mei 2019
Neptunus diluncurkan di Asia Pasifik (Mumbai)	Amazon Neptune sekarang tersedia di Asia Pacific (Mumbai) (ap-south-1).	6 Maret 2019
Posting blog tentang petunjuk kueri Gremlin	Lihat Memperkenalkan petunjuk kueri Gremlin untuk Amazon Neptune.	26 Februari 2019
Neptunus diluncurkan di Asia Pasifik (Tokyo)	Amazon Neptune sekarang tersedia di Asia Pacific (Tokyo) (ap-northeast-1).	23 Januari 2019
CloudFormation template untuk membuat AWS Lambda fungsi untuk mengakses Neptunus	Memperbarui bagian memulai dan menambahk an CloudFormation template untuk membuat fungsi Lambda untuk digunakan dengan Neptunus. Untuk informasi lebih lanjut, lihat Memulai dengan Neptune.	23 Januari 2019

Versi mesin 1.0.1.0.200267.0	Mesin Amazon Neptune versi 1.0.1.0.200267.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200267.0.	21 Januari 2019
Neptunus diluncurkan di Asia Pasifik (Sydney)	Amazon Neptune sekarang tersedia di Asia Pacific (Sydney) (ap-southeast-2).	9 Januari 2019
Posting blog tentang menggunakan Metafaktori	Lihat Menjelajahi Grafik Pengetahuan tentang Amazon Neptune Menggunakan Metaphactory.	9 Januari 2019
Neptunus diluncurkan di Asia Pasifik (Singapura)	Amazon Neptune sekarang tersedia di Asia Pacific (Singapore) (ap-southeast-1).	13 Desember 2018
Versi mesin 1.0.1.0.200264.0	Mesin Amazon Neptune versi 1.0.1.0.200264.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200264.0.	19 November 2018
Dukungan SSL Amazon Neptunus	Sekarang Neptune mendukung koneksi SSL.	19 November 2018
Topik Kesalahan Konsolidasi	Semua pesan kesalahan dan informasi kode sekarang dalam satu topik.	15 November 2018
Topik Memulai Diperbarui	Topik Memulai yang Diperbarui dengan tautan tambahan dan dokumentasi yang diatur ulang.	14 November 2018

Versi mesin 1.0.1.0.200258.0	Mesin Amazon Neptune versi 1.0.1.0.200258.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200258.0.	8 November 2018
Neptunus diluncurkan di Eropa (Frankfurt)	Amazon Neptune sekarang tersedia di Eropa (Frankfurt) (eu-central-1).	7 November 2018
Postingan blog #1 dalam satu seri	Lihat Mari Saya Grafikkan Untuk Anda - Bagian 1 - Rute Udara.	7 November 2018
Posting blog tentang menggunakan Notebook Amazon SageMaker Jupyter	Lihat Menganalisis Grafik Amazon Neptunus menggunakan Notebook Amazon Jupyter. SageMaker	1 November 2018
Versi mesin 1.0.1.0.200255.0	Mesin Amazon Neptune versi 1.0.1.0.200255.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200255.0.	29 Oktober 2018
Neptunus diluncurkan di Eropa (London)	Amazon Neptune sekarang tersedia di Eropa (London) (eu-west-2).	3 Oktober 2018
Versi mesin 1.0.1.0.200237.0	Mesin Amazon Neptune versi 1.0.1.0.200237.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200237.0.	6 September 2018

Versi mesin 1.0.1.0.200236.0	Mesin Amazon Neptune versi 1.0.1.0.200236.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200236.0.	24 Juli 2018
Versi mesin 1.0.1.0.200233.0	Mesin Amazon Neptune versi 1.0.1.0.200233.0 umumnya tersedia. Untuk informasi selengkapnya, lihat Pembaruan 1.0.1.0.200233.0.	22 Juni 2018
Mulai Cepat Neptunus Baru	Diperbarui mulai cepat dengan CloudFormation dan tutorial Konsol Gremlin. Untuk informasi selengkap nya, lihat Amazon Neptunus Mulai Cepat Menggunakan. CloudFormation	19 Juni 2018
Rilis awal Amazon Neptunus	Ini adalah rilis awal Panduan Pengguna Neptune. Lihat juga postingan blog rilis, Amazon Neptune Umumnya Tersedia.	30 Mei 2018
Posting Blog Pengantar Neptunus	Lihat Amazon Neptune - Layanan Database Grafik Terkelola Penuh.	29 November 2017

Memulai dengan Amazon Neptunus

Amazon Neptune adalah layanan basis data grafik yang terkelola penuh yang diskalakan untuk menangani miliaran hubungan dan memungkinkan Anda mengkueri mereka dengan latensi milidetik, dengan biaya rendah untuk jenis kapasitas tersebut.

Jika Anda mencari informasi lebih rinci tentang Neptunus, lihat. Ikhtisar fitur Amazon Neptunus

Jika Anda sudah tahu tentang grafik, lanjutkan ke Menggunakan Neptunus dengan notebook grafik. Atau, jika Anda ingin membuat database Neptunus segera, lihat. Membuat cluster Amazon Neptunus menggunakan AWS CloudFormation

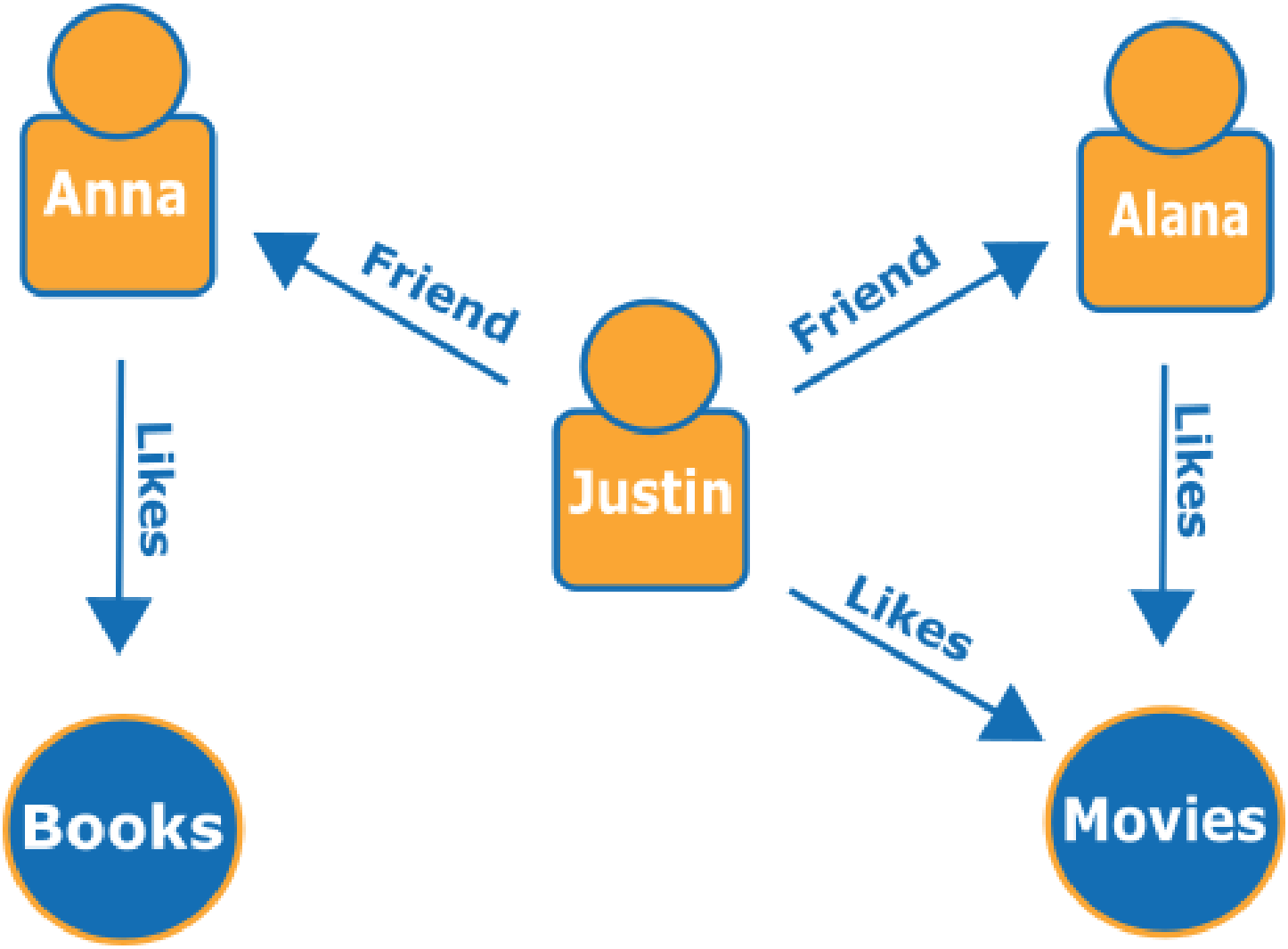
Jika tidak, Anda mungkin ingin tahu lebih banyak tentang database grafik sebelum memulai.

Konsep kunci basis data grafik

Basis data grafik dioptimalkan untuk menyimpan dan mengkueri hubungan antara item-item data.

Basis data ini menyimpan item data sendiri sebagai vertex dari grafik, dan hubungan di antara mereka sebagai edge. Setiap edge memiliki tipe, dan diarahkan dari satu vertex (awal) ke yang lain (akhir). Hubungan bisa disebut predikat serta edge, dan vertex juga kadang-kadang disebut sebagai node. Dalam apa yang disebut grafik properti, baik vertex dan edge dapat memiliki properti tambahan yang terkait dengan keduanya.

Berikut ini adalah grafik kecil yang mewakili teman dan hobi di jejaring sosial:



Edge ditampilkan sebagai panah bernama, dan vertex-vertex mewakili orang-orang tertentu dan hobi yang menghubungkan mereka.

Sebuah traversal sederhana dari grafik ini dapat memberi tahu Anda seperti apa teman Justin.

Mengapa menggunakan basis data grafik?

Setiap kali koneksi atau hubungan antara entitas berada pada inti dari data yang Anda coba buat modelnya, basis data grafik adalah pilihan alami Anda.

Untuk satu hal, mudah untuk membuat model interkoneksi data sebagai grafik, dan kemudian menulis kueri kompleks yang mengekstrak informasi dunia nyata dari grafik.

Membangun aplikasi setara menggunakan basis data relasional mengharuskan Anda untuk membuat banyak tabel dengan beberapa kunci asing dan kemudian menulis kueri SQL nested dan kompleks

digabungkan. Pendekatan itu tidak hanya cepat menjadi berat dari perspektif pengkodean, kinerjanya terdegradasi dengan cepat karena jumlah data meningkat.

Sebaliknya, basis data grafik seperti Neptune dapat mengajukan kueri untuk hubungan antara miliaran vertex tanpa membuat macet.

Apa yang dapat Anda lakukan dengan basis data grafik?

Grafik dapat mewakili keterkaitan entitas dunia nyata dalam banyak hal, dalam hal tindakan, kepemilikan, orang tua, pilihan pembelian, koneksi pribadi, ikatan keluarga, dan sebagainya.

Berikut adalah beberapa area yang paling umum tempat basis data grafik digunakan:

- Grafik pengetahuan — Grafik pengetahuan memungkinkan Anda mengatur dan menanyakan semua jenis informasi yang terhubung untuk menjawab pertanyaan umum. Menggunakan grafik pengetahuan, Anda dapat menambahkan informasi topikal untuk katalog produk, dan informasi beragam model seperti yang terkandung dalam Wikidata.

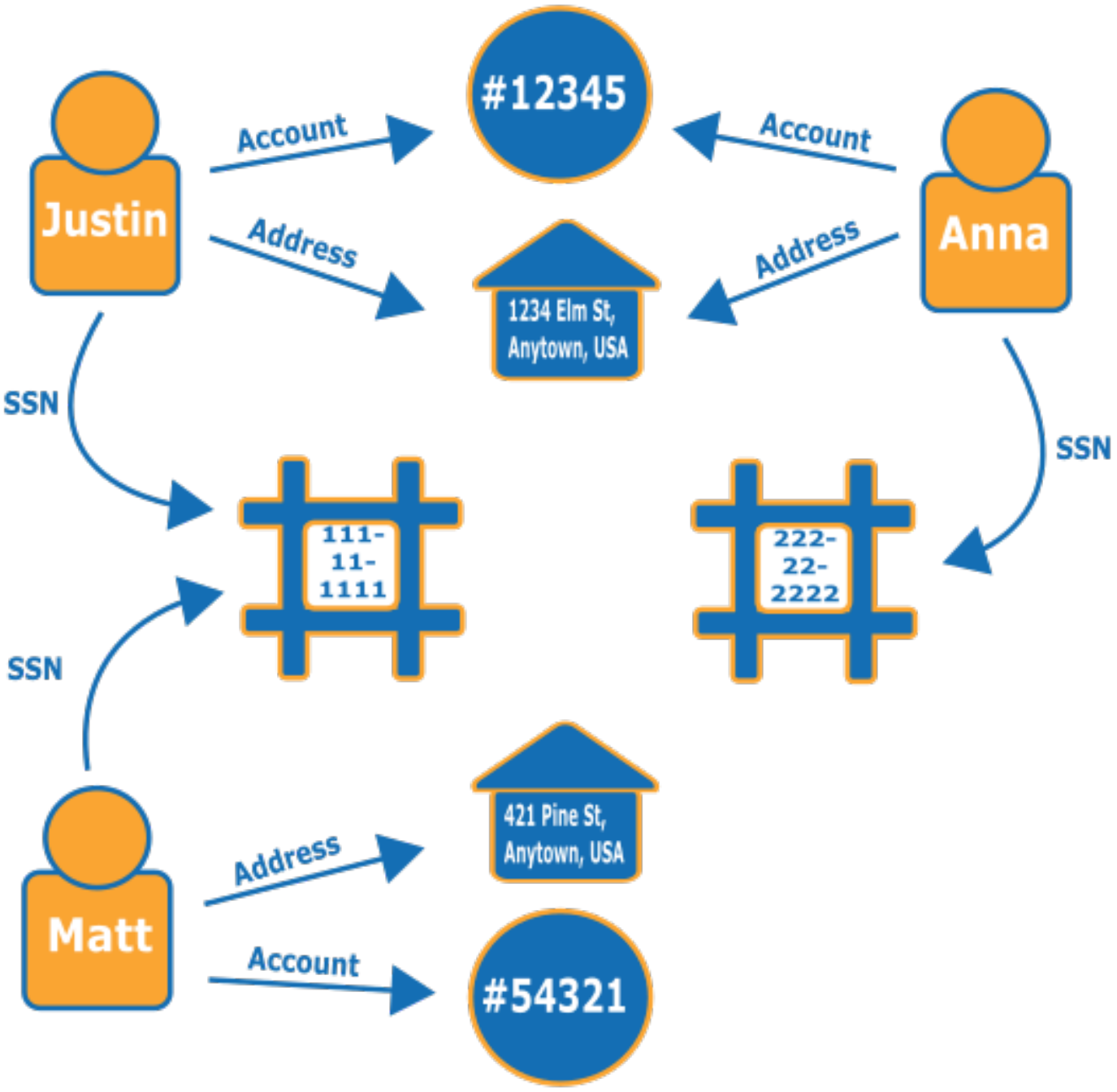
  Untuk mempelajari lebih lanjut tentang bagaimana grafik pengetahuan bekerja dan di mana mereka sedang digunakan, lihat Grafik Pengetahuan di AWS.
- Grafik identitas — Dalam basis data grafik, Anda dapat menyimpan hubungan antara kategori informasi seperti minat pelanggan, teman, dan riwayat pembelian, dan kemudian mengajukan kueri untuk data tersebut untuk membuat rekomendasi yang dipersonalisasi dan relevan.

  Misalnya, Anda dapat menggunakan basis data grafik untuk membuat rekomendasi produk kepada pengguna berdasarkan produk mana yang dibeli oleh orang lain yang mengikuti olahraga yang sama dan memiliki riwayat pembelian yang sama. Atau, Anda dapat mengidentifikasi orang yang memiliki teman yang sama tetapi belum mengenal satu sama lain, dan membuat rekomendasi persahabatan.

  Grafik semacam ini dikenal sebagai grafik identitas, dan secara luas digunakan untuk personalisasi interaksi dengan pengguna. Untuk mengetahui lebih lanjut, lihat Grafik Identitas di AWS. Untuk memulai membangun grafik identitas Anda sendiri, Anda dapat mulai dengan sampel Grafik Identitas Menggunakan Amazon Neptune.
- Grafik Penipuan — Ini adalah penggunaan umum untuk basis data grafik. Basis data dapat membantu Anda melacak pembelian kartu kredit dan lokasi pembelian untuk mendeteksi penggunaan yang tidak biasa, atau untuk mendeteksi pembeli yang mencoba menggunakan alamat email dan kartu kredit yang sama seperti yang digunakan dalam kasus penipuan yang diketahui. Mereka dapat mengizinkan Anda memeriksa beberapa orang yang terkait dengan

alamat email pribadi, atau beberapa orang di lokasi fisik berbeda yang berbagi alamat IP yang sama.

Pertimbangkan grafik berikut. Grafik ini menunjukkan hubungan tiga orang dan informasi terkait identitas mereka. Setiap orang memiliki alamat, rekening bank, dan nomor jaminan sosial. Namun, kita dapat melihat bahwa Matt dan Justin berbagi nomor jaminan sosial yang sama, yang tidak biasa dan menunjukkan kemungkinan penipuan oleh salah satu dari mereka. Kueri untuk grafik penipuan dapat mengungkapkan koneksi semacam ini sehingga mereka dapat ditinjau.



Untuk mempelajari lebih lanjut tentang grafik penipuan dan di mana mereka sedang digunakan, lihat Grafik Penipuan di AWS.

- Jaringan sosial — Salah satu bidang pertama dan paling umum tempat basis data grafik digunakan adalah dalam aplikasi jejaring sosial.

Misalnya, anggaplah Anda ingin membangun umpan sosial ke dalam suatu situs web. Anda dapat dengan mudah menggunakan basis data grafik di ujung belakang untuk memberikan hasil kepada pengguna yang mencerminkan pembaruan terbaru dari keluarga mereka, teman mereka, dari orang-orang yang pembaruannya mereka “sukai”, dan dari orang-orang yang tinggal dekat dengan mereka.

- Petunjuk arah mengemudi — Grafik dapat membantu menemukan rute terbaik dari titik awal ke tujuan, mempertimbangkan lalu lintas saat ini dan pola lalu lintas yang khas.
- Logistik — Grafik dapat membantu mengidentifikasi cara paling efisien untuk menggunakan sumber daya pengiriman dan distribusi yang tersedia untuk memenuhi kebutuhan pelanggan.
- Diagnosis — Grafik dapat mewakili pohon diagnostik kompleks yang dapat diajukan kueri untuk mengidentifikasi sumber masalah dan kegagalan yang diamati.
- Penelitian ilmiah — Dengan database grafik, Anda dapat membangun aplikasi yang menyimpan dan menavigasi data ilmiah dan bahkan informasi medis sensitif menggunakan enkripsi saat istirahat. Misalnya, Anda bisa menyimpan model interaksi penyakit dan gen. Anda dapat mencari pola grafik dalam jalur protein untuk menemukan gen lain yang mungkin terkait dengan suatu penyakit. Anda dapat membuat model senyawa kimia sebagai grafik dan mengajukan kueri untuk pola dalam struktur molekul. Anda dapat mengorelasikan data pasien dari rekam medis dalam sistem yang berbeda. Anda dapat secara topikal mengatur publikasi penelitian untuk menemukan informasi yang relevan dengan cepat.
- Aturan peraturan — Anda dapat menyimpan persyaratan peraturan yang kompleks sebagai grafik, dan menanyakannya untuk mendeteksi situasi di mana mereka mungkin berlaku untuk operasi day-to-day bisnis Anda.
- Topologi dan peristiwa jaringan — Basis data grafik dapat membantu Anda mengelola dan melindungi jaringan IT. Ketika Anda menyimpan topologi jaringan sebagai grafik, Anda juga dapat menyimpan dan memproses berbagai jenis peristiwa di jaringan. Anda dapat menjawab pertanyaan seperti berapa banyak host yang menjalankan aplikasi tertentu. Anda dapat mengajukan kueri untuk pola yang mungkin menunjukkan bahwa host tertentu telah diganggu oleh program berbahaya, dan kueri untuk koneksi data yang dapat membantu melacak program ke host asli yang mengunduhnya itu.

Bagaimana Anda menanyakan grafik?

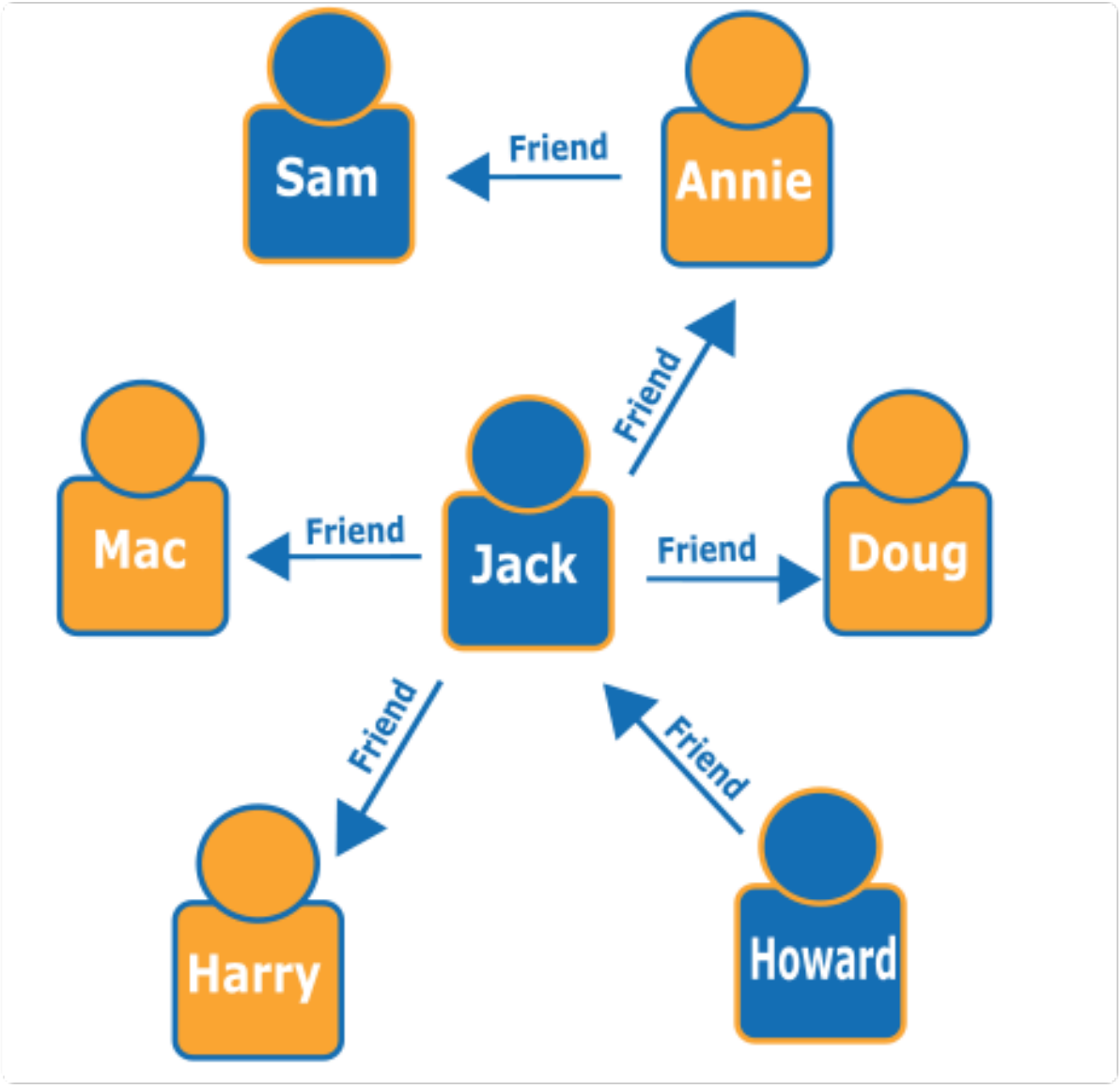
Neptunus mendukung tiga bahasa query tujuan khusus yang dirancang untuk query data grafik dari berbagai jenis. Anda dapat menggunakan bahasa-bahasa ini untuk menambah, memodifikasi, menghapus, dan menanyakan data dalam database grafik Neptunus:

- Gremlin adalah bahasa traversal grafik untuk grafik properti. Sebuah kueri di Gremlin adalah sebuah traversal yang terdiri dari langkah-langkah berlainan, yang masing-masing mengikuti edge ke simpul. Lihat dokumentasi Gremlin di Apache TinkerPop untuk informasi lebih lanjut.

  Implementasi Neptune dari Gremlin memiliki beberapa perbedaan dari implementasi lain, terutama ketika Anda menggunakan Gremlin-Groovy (kueri Gremlin dikirim sebagai teks serial). Untuk informasi selengkapnya, lihat Kepatuhan standar Gremlin di Amazon Neptune.
- OpenCypherOpenCypher adalah bahasa query deklaratif untuk grafik properti yang awalnya dikembangkan oleh Neo4j, kemudian open-source pada tahun 2015, dan berkontribusi pada proyek OpenCypher di bawah lisensi open-source Apache 2. Lihat Referensi Bahasa Kueri Cypher (Versi 9) untuk spesifikasi bahasa, serta Panduan Gaya Cypher untuk informasi tambahan.
- SPARQL adalah bahasa query deklaratif untuk data RDF, berdasarkan pencocokan pola grafik yang distandarisasi oleh World Wide Web Consortium (W3C) dan dijelaskan dalam SPARQL 1.1 Ikhtisar) dan spesifikasi SPARQL 1.1 Query Language. Lihat Kepatuhan standar SPARQL di Amazon Neptunus untuk detail spesifik tentang implementasi Neptunus SPARQL.

Contoh pencocokan kueri Gremlin dan SPARQL

Mengingat grafik orang (node) berikut dan hubungan mereka (edge), Anda dapat mengetahui siapa “teman dari teman” dari orang tertentu - misalnya, teman-teman dari teman-temannya Howard.



Melihat grafiknya, Anda dapat melihat bahwa Howard memiliki satu teman, Jack, dan Jack memiliki empat teman: Annie, Harry, Doug, dan Mac. Ini adalah contoh sederhana dengan grafik sederhana, tetapi jenis kueri ini dapat diskalakan dalam kompleksitas, ukuran dataset, dan ukuran hasil.

Berikut ini adalah kueri traversal Gremlin yang mengembalikan nama-nama teman-teman dari teman-teman Howard:

```
g.V().has('name', 'Howard').out('friend').out('friend').values('name')
```

Berikut ini adalah kueri SPARQL yang mengembalikan nama-nama teman-teman dari teman-teman Howard:

```
prefix : <#>

  select ?names where {
    ?howard :name "Howard" .
    ?howard :friend/:friend/:name ?names .
  }
```

> **Note**
>
> Setiap bagian dari triple Resource Description Framework (RDF) mana pun memiliki URI terkait dengan bagian tersebut. Dalam contoh di atas, prefiks URI sengaja singkat.

Ikuti kursus online tentang menggunakan Amazon Neptune

Jika Anda suka belajar dengan video, AWS menawarkan kursus online di AWS Online Tech Talks untuk membantu Anda memulai. Salah satu yang memperkenalkan database grafik adalah:

Pengenalan Database Grafik, penyelaman mendalam, dan demo dengan Amazon Neptunus.

Menggali lebih dalam arsitektur referensi grafik

Ketika Anda berpikir tentang masalah apa yang dapat dipecahkan oleh database grafik untuk Anda, dan bagaimana mendekatinya, salah satu tempat terbaik untuk memulai adalah proyek arsitektur referensi grafik Neptunus. GitHub

Di sana Anda dapat menemukan penjelasan detail tentang jenis beban kerja grafik, dan tiga bagian untuk membantu Anda merancang basis data grafik yang efektif:

- Model Data dan Bahasa Kueri — Bagian ini memandu Anda melalui perbedaan antara Gremlin dan SPARQL dan bagaimana memilih di antara mereka.
- Pemodelan Data Grafik — Ini adalah diskusi menyeluruh tentang bagaimana membuat keputusan pemodelan data grafik, termasuk panduan detail pemodelan grafik properti menggunakan pemodelan Gremlin dan RDF menggunakan SPARQL.
- Mengonversi Model Data Lain ke Model Grafik — Di sini Anda dapat mengetahui bagaimana cara menerjemahkan model data relasional ke dalam model grafik.

Ada juga tiga bagian yang memandu Anda melalui langkah-langkah khusus untuk menggunakan Neptune:

- Menghubungkan ke Amazon Neptune dari Klien di Luar VPC Neptune — Bagian ini menunjukkan beberapa pilihan untuk menghubungkan ke Neptune dari luar VPC tempat klaster DB Anda berada.
- Mengakses Amazon Neptunus dari Fungsi Lambda - Di sini Anda akan mengetahui cara menghubungkan dengan andal ke AWS Neptunus dari fungsi Lambda.
- Menulis ke Amazon Neptune dari Amazon Kinesis Data Stream — Bagian ini dapat membantu Anda menangani skenario throughput tulis tinggi dengan Neptune.

Menggunakan Amazon Neptunus dengan notebook grafik

Untuk bekerja dengan grafik Neptunus, Anda dapat menggunakan buku catatan grafik Neptunus atau membuat database Neptunus baru menggunakan templat. CloudFormation

Baik Anda baru mengenal grafik dan ingin belajar dan bereksperimen, atau Anda berpengalaman dan ingin menyempurnakan kueri Anda, meja kerja Neptunus menawarkan lingkungan pengembangan interaktif (IDE) yang dapat meningkatkan produktivitas Anda saat Anda membuat aplikasi grafik. Workbench menyediakan antarmuka yang mudah digunakan untuk berinteraksi dengan database Neptunus Anda, menulis kueri, dan memvisualisasikan data Anda.

Dengan menggunakan CloudFormation template untuk mengatur database Neptunus Anda, dan Workbench untuk mengembangkan aplikasi grafik Anda, Anda dapat memulai dengan Neptunus dengan cepat dan efisien, tanpa perlu perkakas tambahan. Ini memungkinkan Anda untuk fokus pada membangun aplikasi Anda daripada menyiapkan infrastruktur yang mendasarinya.

> **Note**
>
> Notebook Neptunus, yang dikelola melalui SageMaker Amazon AI, saat ini tidak tersedia di wilayah Asia Pasifik (Malaysia) (ap-tenggara 5). Namun, Anda masih dapat menggunakan notebook Neptunus melalui opsi alternatif yang tidak dikelola. Lihat Menyiapkan notebook Neptunus secara manual untuk menyebarkan buku catatan secara manual.

Neptunus menyediakan Jupyter dan notebook dalam proyek JupyterLabnotebook grafik Neptunus open-source, dan di meja kerja Neptunus. GitHub Notebook ini menawarkan contoh tutorial aplikasi dan cuplikan kode dalam lingkungan pengkodean interaktif di mana Anda dapat belajar tentang

teknologi grafik dan Neptunus. Anda dapat menggunakannya untuk memandu Anda mengatur, mengonfigurasi, mengisi, dan mengajukan kueri grafik menggunakan bahasa kueri yang berbeda, set data yang berbeda, dan bahkan basis data yang berbeda di ujung belakang.

Anda dapat meng-host notebook ini dengan beberapa cara berbeda:

- Meja kerja Neptunus memungkinkan Anda menjalankan notebook Jupyter di lingkungan yang dikelola sepenuhnya, dihosting di SageMaker Amazon AI, dan secara otomatis memuat rilis terbaru proyek notebook grafik Neptunus untuk Anda. Sangat mudah untuk mengatur meja kerja di konsol Neptunus saat Anda membuat database Neptunus baru.

  > **Note**
  >
  > Saat membuat instance notebook Neptunus, Anda diberikan dua opsi untuk akses jaringan: Akses langsung melalui SageMaker Amazon AI (default) dan akses melalui VPC. Dalam kedua opsi, notebook memerlukan akses ke internet untuk mengambil dependensi paket untuk menginstal meja kerja Neptunus. Kurangnya akses internet akan menyebabkan pembuatan instance notebook Neptunus gagal.

- Anda juga dapat menginstal Jupyter secara lokal. Ini memungkinkan Anda menjalankan notebook dari laptop Anda, terhubung ke Neptunus atau ke instance lokal dari salah satu database grafik sumber terbuka. Dalam kasus terakhir, Anda dapat bereksperimen dengan teknologi grafik sebanyak yang Anda inginkan sebelum Anda menghabiskan satu sen. Kemudian, ketika Anda siap, Anda dapat bergerak dengan lancar ke lingkungan produksi terkelola yang ditawarkan Neptunus.

Menggunakan workbench Neptune untuk meng-host notebook Neptune

Neptunus T3 menawarkan T4g dan jenis instans yang dapat Anda mulai dengan harga kurang dari $0,10 per jam. Anda ditagih untuk sumber daya meja kerja melalui Amazon SageMaker AI, terpisah dari penagihan Neptunus Anda. Lihat halaman harga Neptunus. Jupyter dan JupyterLab notebook yang dibuat di meja kerja Neptunus semuanya menggunakan lingkungan Amazon Linux 2 dan 4. JupyterLab Untuk informasi selengkapnya tentang dukungan JupyterLab notebook, lihat dokumentasi Amazon SageMaker AI.

Anda dapat membuat Jupyter atau JupyterLab notebook menggunakan meja kerja Neptunus dengan salah satu dari dua cara: Konsol Manajemen AWS

- Gunakan menu konfigurasi Notebook saat membuat cluster DB Neptunus baru. Untuk melakukan ini, ikuti langkah-langkah yang diuraikan. Meluncurkan cluster DB Neptunus menggunakan Konsol Manajemen AWS
- Gunakan menu Notebooks di panel navigasi kiri setelah cluster DB Anda telah dibuat. Untuk melakukan ini, ikuti langkah-langkah di bawah ini.

Untuk membuat Jupyter atau JupyterLab notebook menggunakan menu Notebooks

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Pada panel navigasi di sebelah kiri, pilih Notebook.
3. Pilih Buat Notebook.
4. Pilih Database sebagai layanan Neptunus.
5. Di daftar Klaster, pilih nama klaster DB Neptune Anda. Jika Anda belum memiliki klaster DB, pilih Buat klaster untuk membuatnya.
6. Pilih jenis instans Notebook.
7. Beri nama notebook Anda, dan deskripsinya bila mau.
8. Kecuali Anda telah membuat peran AWS Identity and Access Management (IAM) untuk buku catatan Anda, pilih Buat peran IAM, dan masukkan nama peran IAM.

   > **Note**
   >
   > Jika Anda memilih untuk menggunakan kembali peran IAM yang dibuat untuk buku catatan sebelumnya, kebijakan peran harus berisi izin yang benar untuk mengakses kluster DB Neptunus yang Anda gunakan. Anda dapat memverifikasi ini dengan memeriksa apakah komponen dalam ARN sumber daya di bawah `neptune-db:*` tindakan cocok dengan cluster tersebut. Izin yang tidak dikonfigurasi dengan benar mengakibatkan kesalahan koneksi saat Anda mencoba menjalankan perintah ajaib notebook.

9. Pilih Buat Notebook. Proses pembuatan mungkin memakan waktu 5 hingga 10 menit sebelum semuanya siap.
10. Setelah buku catatan Anda dibuat, pilih dan kemudian pilih Buka Jupyter atau Buka. JupyterLab

Konsol dapat membuat (IAM) role AWS Identity and Access Management untuk notebook Anda, atau Anda dapat membuatnya sendiri. Kebijakan untuk peran ini harus mencakup berikut ini:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowS3GetObjectS3ListBucket",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::aws-neptune-notebook",
        "arn:aws:s3:::aws-neptune-notebook/*",
        "arn:aws:s3:::aws-neptune-notebook-us-east-1",
        "arn:aws:s3:::aws-neptune-notebook-us-east-1/*"
      ]
    },
    {
      "Sid": "AllowAccessNeptuneDB",
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": [
        "arn:aws:neptune-db:us-east-1:111122223333:cluster-resource-id/*"
      ]
    }
  ]
}
```

Perhatikan bahwa pernyataan kedua dalam kebijakan di atas mencantumkan satu atau lebih sumber daya cluster Neptunus. IDs

Selain itu, peran harus menetapkan hubungan kepercayaan berikut:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Sekali lagi, menyiapkan semuanya bisa memakan waktu 5 hingga 10 menit.

Anda dapat mengonfigurasi notebook baru Anda agar berfungsi dengan Neptunus ML, seperti yang dijelaskan di. Mengkonfigurasi notebook Neptunus secara manual untuk Neptunus ML

Menggunakan Python untuk menghubungkan notebook SageMaker AI generik ke Neptunus

Menghubungkan notebook ke Neptunus mudah jika Anda telah menginstal sihir Neptunus, tetapi juga memungkinkan untuk menghubungkan SageMaker notebook AI ke Neptunus menggunakan Python, bahkan jika Anda tidak menggunakan notebook Neptunus.

Langkah-langkah yang harus diambil untuk terhubung ke Neptunus di SageMaker sel notebook AI

1. Instal klien Python Gremlin:

   ```
   !pip install gremlinpython
   ```

   Notebook Neptunus menginstal klien Python Gremlin untuk Anda, jadi langkah ini hanya diperlukan jika Anda menggunakan notebook AI biasa. SageMaker
2. Tulis kode seperti berikut ini untuk menghubungkan dan mengeluarkan kueri Gremlin:

   ```
   from gremlin_python import statics
   from gremlin_python.structure.graph import Graph
   ```

```
from gremlin_python.process.graph_traversal import __
from gremlin_python.process.strategies import *
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection
from gremlin_python.driver.aiohttp.transport import AiohttpTransport
from gremlin_python.process.traversal import *
import os

port = 8182
server = '(your server endpoint)'

endpoint = f'wss://{server}:{port}/gremlin'

graph=Graph()

connection = DriverRemoteConnection(endpoint,'g',

 transport_factory=lambda:AiohttpTransport(call_from_event_loop=True))

g = graph.traversal().withRemote(connection)

results = (g.V().hasLabel('airport')
                .sample(10)
                .order()
                .by('code')
                .local(__.values('code','city').fold())
                .toList())

# Print the results in a tabular form with a row index
for i,c in enumerate(results,1):
    print("%3d %4s %s" % (i,c[0],c[1]))

connection.close()
```

Note

Jika Anda kebetulan menggunakan versi klien Python Gremlin yang lebih tua dari 3.5.0, baris ini:

```
connection = DriverRemoteConnection(endpoint,'g',

 transport_factory=lambda:AiohttpTransport(call_from_event_loop=True))
```

Hanya akan menjadi:

```
connection = DriverRemoteConnection(endpoint,'g')
```

Meluncurkan grafik-notebook sebagai Amazon Neptune Workbench via CloudFormation

Anda dapat menerapkan notebook meja kerja Amazon Neptunus menggunakan. CloudFormation CloudFormation Template menyebarkan notebook meja kerja Neptunus sebagai sumber daya dan menyertakan notebook dasar 'Memulai'. Meja kerja memungkinkan Anda bekerja dengan kluster Database Amazon Neptunus menggunakan notebook Jupyter yang dihosting oleh Amazon. SageMaker Anda ditagih untuk sumber daya meja kerja melalui Amazon SageMaker, terpisah dari penagihan Neptunus Anda.

CloudFormation Templat dan instruksi terperinci, termasuk persyaratan peran dan izin IAM, tersedia di repositori GitHub grafik-notebook.

Mengaktifkan CloudWatch log pada Notebook Neptunus

CloudWatch log sekarang diaktifkan secara default untuk Notebook Neptunus. Jika Anda memiliki buku catatan lama yang tidak menghasilkan CloudWatch log, ikuti langkah-langkah berikut untuk mengaktifkannya secara manual:

1. Masuk ke Konsol Manajemen AWS dan buka konsol SageMaker AI.
2. Pada panel navigasi di sebelah kiri, pilih Notebook, lalu Instans Notebook. Cari nama notebook Neptunus yang ingin Anda aktifkan log.
3. Buka halaman detail dengan memilih nama instance notebook itu.
4. Jika instance notebook sedang berjalan, pilih tombol Stop, di kanan atas halaman detail notebook.
5. Di bawah Izin dan enkripsi ada bidang untuk peran IAM ARN. Pilih tautan di bidang ini untuk pergi ke peran IAM yang dijalankan oleh instance notebook ini.
6. Buat kebijakan berikut:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DeleteLogDelivery",
        "logs:Describe*",
        "logs:GetLogDelivery",
        "logs:GetLogEvents",
        "logs:ListLogDeliveries",
        "logs:PutLogEvents",
        "logs:PutResourcePolicy",
        "logs:UpdateLogDelivery"
      ],
      "Resource": "*"
    }
  ]
}
```

7. Simpan kebijakan baru ini dan lampirkan ke Peran IAM yang ditemukan di Langkah 4.
8. Klik Mulai di kanan atas halaman detail instans notebook SageMaker AI.
9. Saat log mulai mengalir, Anda akan melihat tautan Lihat Log di bawah bidang berlabel konfigurasi Siklus Hidup di dekat kiri bawah bagian pengaturan instans Notebook pada halaman detail.

Jika notebook gagal memulai, akan ada pesan dari halaman detail notebook di konsol SageMaker AI, yang menyatakan bahwa instance notebook membutuhkan waktu lebih dari 5 menit untuk memulai. CloudWatch log yang relevan dengan masalah ini dapat ditemukan dengan nama ini:

```
(your-notebook-name)/LifecycleConfigOnStart
```

Menyiapkan notebook grafik di komputer lokal Anda

Proyek grafik-notebook memiliki instruksi untuk menyiapkan notebook Neptune di mesin lokal Anda:

- Prasyarat
- Jupyter dan instalasi JupyterLab
- Menghubungkan ke basis data gafik.

Anda dapat menghubungkan notebook lokal Anda baik ke klaster DB Neptune, atau ke instans lokal atau remote dari basis data grafik sumber terbuka.

Menggunakan notebook Neptune dengan klaster Neptune

Jika Anda terhubung ke cluster Neptunus di bagian belakang, Anda mungkin ingin menjalankan notebook di Amazon AI. SageMaker Menghubungkan ke Neptunus SageMaker dari AI bisa lebih nyaman daripada dari instalasi notebook lokal, dan itu akan memungkinkan Anda bekerja lebih mudah dengan Neptunus ML.

Untuk petunjuk tentang cara mengatur notebook di SageMaker AI, lihat Meluncurkan grafik-notebook menggunakan Amazon. SageMaker

Untuk petunjuk tentang cara mengatur dan mengonfigurasi Neptune sendiri, lihat Menyiapkan Amazon Neptunus.

Anda juga dapat menghubungkan instalasi lokal notebook Neptune ke klaster DB Neptune. Ini bisa agak lebih rumit karena klaster DB Amazon Neptune hanya dapat dibuat di Amazon Virtual Private Cloud (VPC), yang secara desain terisolasi dari dunia luar. Ada beberapa cara untuk terhubung ke VPC dari luarnya. Salah satunya adalah menggunakan penyeimbang beban. Lainnya adalah dengan menggunakan peering VPC (lihat Panduan Peering Amazon Virtual Private Cloud).

Namun, cara yang paling mudah bagi kebanyakan orang adalah dengan menyambung untuk menyediakan server proksi Amazon EC2 dalam VPC dan kemudian menggunakan Terowongan SSH (juga dipanggil penerusan port), untuk menyambung kepadanya. Anda dapat menemukan petunjuk tentang cara mengatur di Menghubungkan notebook grafik secara lokal ke Amazon Neptunusadditional-databases/neptune di folder proyek grafik-notebook. GitHub

Menggunakan notebook Neptune dengan basis data grafik sumber terbuka

Untuk memulai dengan teknologi grafik tanpa biaya, Anda juga dapat menggunakan notebook Neptune dengan berbagai basis data sumber terbuka di ujung belakang. Contohnya adalah server TinkerPop Gremlin, dan database Blazegraph.

Untuk menggunakan Gremlin Server sebagai basis data back-end Anda, ikuti langkah-langkah berikut:

- Menghubungkan grafik-notebook ke folder Server Gremlin. GitHub
- Folder konfigurasi Gremlin grafik-notebook. GitHub

Untuk menggunakan instance lokal Blazegraph sebagai basis data back-end Anda, ikuti langkah-langkah berikut:

- Tinjau instruksi mulai cepat Blazegraph untuk memahami pengaturan dasar dan konfigurasi yang diperlukan untuk menjalankan instance Blazegraph.
- Akses GitHub folder konfigurasi Blazegraph grafik-notebook yang berisi file dan instruksi yang diperlukan untuk menyiapkan instance Blazegraph lokal.
- Di dalam GitHub repositori, arahkan ke direktori “blazegraph” dan ikuti instruksi yang diberikan untuk menyiapkan instance Blazegraph lokal Anda. Ini termasuk langkah-langkah untuk mengunduh perangkat lunak Blazegraph, mengonfigurasi file yang diperlukan, dan memulai server Blazegraph.

Setelah menjalankan instance Blazegraph lokal, Anda dapat mengintegrasikannya dengan aplikasi Anda sebagai basis data backend untuk data dan kueri berbasis grafik Anda. Lihat dokumentasi dan kode contoh yang disediakan di repositori grafik-notebook untuk mempelajari cara menghubungkan aplikasi Anda ke instance Blazegraph.

Migrasi notebook Neptunus ke 4.x JupyterLab

Bagian ini menguraikan berbagai pendekatan untuk memigrasikan notebook Neptunus Anda ke JupyterLab 4.x dan lingkungan Amazon Linux yang lebih baru. Untuk informasi rinci tentang pembuatan versi, lihat Amazon SageMaker AI JupyterLab JupyterLab Versioning.

Pendekatan migrasi

Instalasi baru

Jika Anda tidak perlu menyimpan file atau konfigurasi ruang kerja yang ada, Anda dapat:

1. Buat instance notebook baru yang menjalankan JupyterLab 4.x (notebook-al2-v3)
2. Verifikasi pengaturan baru berfungsi seperti yang diharapkan
3. Hentikan dan hapus instance notebook lama Anda

Migrasi transfer file

Metode ini menggunakan sistem lokal Anda atau Amazon S3 sebagai penyimpanan perantara.

Terbaik untuk

- Akses internet langsung melalui konfigurasi jaringan Amazon SageMaker AI.
- Volume data moderat untuk bermigrasi
- File khusus untuk diawetkan, bukan seluruh konfigurasi ruang kerja.

Metode 1: Menggunakan JupyterLab UI

Terbaik untuk

- Sejumlah kecil file
- Migrasi file selektif
- Lebih suka drag-and-drop operasi sederhana

Langkah-langkah

1. Unduh file dari JupyterLab contoh sumber:
   - Navigasikan dan pilih file yang ingin Anda migrasikan ke instance baru JupyterLab
   - Klik kanan dan pilih Unduh
2. Unggah ke JupyterLab instance baru:
   - Gunakan tombol unggah JupyterLab dan pilih semua file yang ingin Anda salin ke instance baru

- (Atau) seret dan lepas file secara langsung

Metode 2: Menggunakan Amazon S3

Terbaik untuk

- Sejumlah besar file
- Melestarikan struktur folder Anda
- Migrasi massal

Prasyarat

Pastikan peran yang terkait dengan notebook memiliki izin yang sesuai untuk mengunggah dan mengakses bucket Amazon S3:

```
{
"Effect": "Allow",
"Action": ["s3:PutObject", "s3:GetObject", "s3:ListBucket"],
"Resource": ["arn:aws:s3:::your-bucket-name/*", "arn:aws:s3:::your-bucket-name"]
}
```

> **Note**
>
> AWS CLI harus diinstal sebelumnya pada SageMaker notebook AI.

Langkah-langkah

1. Buka terminal JupyterLab atau ketik perintah terminal di sel notebook dengan ! awalan.
2. Salin file dari JupyterLab instance lama Anda ke S3 menggunakan perintah CLI sinkronisasi Amazon S3 cp atau Amazon S3:

```
# using AWS s3 cp
aws s3 cp /home/ec2-user/SageMaker/your-folder s3://your-bucket/backup/ --recursive

# (OR) using AWS s3 sync
aws s3 sync /home/ec2-user/SageMaker/your-folder s3://your-bucket/backup/
```

3. Salin file dari S3 ke JupyterLab instance baru Anda:

```
# using AWS s3 cp
aws s3 cp s3://your-bucket/backup/ /home/ec2-user/SageMaker/your-folder --recursive

# (OR) using AWS s3 sync
aws s3 sync s3://your-bucket/backup/ /home/ec2-user/SageMaker/your-folder
```

> **Note**
>
> Gunakan `sync` untuk memelihara struktur folder dan pembaruan tambahan dan `cp` untuk transfer satu kali.

Migrasi Amazon EFS

Terbaik untuk

- Konfigurasi jaringan khusus VPC
- Volume data yang besar

Langkah-langkah

Ikuti sistem file Mount an EFS ke blog notebook Amazon SageMaker AI untuk menggunakan sistem file Amazon EFS dengan instans notebook Anda.

Selain itu, ada juga beberapa langkah lagi yang berlaku khusus untuk memigrasikan notebook Neptunus ke lingkungan baru:

1. Selama pembuatan notebook Neptunus di konsol, pilih Buat konfigurasi siklus hidup baru di bawah Konfigurasi Siklus Hidup
2. Dalam konfigurasi siklus hidup template, tambahkan perintah mount `sudo mount -t nfs ...` () Amazon EFS Anda setelah skrip install.sh

Ini memastikan sistem file Amazon EFS Anda dipasang secara otomatis setiap kali instans notebook Anda dimulai atau dimulai ulang. Untuk mengatasi masalah pemasangan, lihat dokumen pemecahan masalah Amazon EFS.

Keuntungan

- Akses tanpa batas ke file di seluruh instance
- Akses file langsung tanpa transfer perantara
- Penanganan kumpulan data besar yang efisien

Migrasi volume Amazon EBS

Terbaik untuk saat Anda perlu melestarikan

- Konfigurasi ruang kerja lengkap
- File tersembunyi
- Pengaturan sistem
- Melestarikan konfigurasi ruang kerja yang lengkap, file tersembunyi, dan pengaturan sistem

Ikuti panduan migrasi AWS SageMaker AI untuk volume Amazon EBS untuk mentransfer file dari volume Amazon EBS yang terkait dengan instans notebook.

Selain itu, ada juga beberapa langkah lagi yang berlaku khusus untuk memigrasikan notebook Neptunus ke lingkungan baru.

Prasyarat khusus Neptunus

Di peran IAM notebook Neptunus sumber, tambahkan semua izin berikut:

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:ListBucket",
    "s3:CreateBucket",
    "s3:PutObject"
  ],
  "Resource": [
    "arn:aws:s3:::(your ebs backup bucket name)",
    "arn:aws:s3:::(your ebs backup bucket name)/*"
  ]
```

```
},
{
  "Effect": "Allow",
  "Action": [
    "sagemaker:ListTags"
  ],
  "Resource": [
    "*"
  ]
}
```

Pastikan untuk menentukan ARN yang benar untuk bucket S3 yang akan Anda gunakan untuk membuat cadangan.

Konfigurasi siklus hidup khusus Neptunus

Saat membuat skrip konfigurasi Siklus Hidup kedua untuk memulihkan cadangan (`darion-create.sh`) seperti yang dijelaskan dalam posting blog, nama Siklus Hidup harus mengikuti format, seperti. `aws-neptune-*` `aws-neptune-sync-from-s3` Ini memastikan bahwa LCC dapat dipilih selama pembuatan notebook di konsol Neptunus.

Sinkronisasi khusus Neptunus dari snapshot ke instance baru

Dalam langkah-langkah yang dijelaskan dalam posting blog untuk menyinkronkan dari snapshot ke instance baru, berikut adalah perubahan khusus Neptunus:

- Pada langkah 4, pilih notebook-al2-v3.
- Pada langkah 5, gunakan kembali peran IAM dari sumber notebook Neptunus.
- Antara langkah 7 dan 8:
  - Dalam pengaturan instance Notebook, tetapkan nama yang menggunakan `aws-neptune-*` format.
  - Buka akordeon pengaturan Jaringan dan pilih grup VPC, Subnet, dan Keamanan yang sama seperti di notebook sumber.

Langkah-langkah khusus Neptunus setelah notebook baru dibuat

1. Pilih tombol Open Jupyter untuk notebook. Setelah `SYNC_COMPLETE` file muncul di direktori utama, lanjutkan ke langkah berikutnya.

2. Buka halaman instance notebook di konsol SageMaker AI.
3. Hentikan notebook.
4. Pilih Edit.
5. Di pengaturan instance notebook, edit bidang konfigurasi Siklus Hidup dengan memilih Siklus Hidup asli notebook Neptunus sumber. Perhatikan bahwa ini bukan Siklus Hidup cadangan EBS.
6. Pilih Perbarui pengaturan buku catatan.
7. Mulai notebook lagi.

Dengan modifikasi yang dijelaskan di sini untuk langkah-langkah yang diuraikan dalam posting blog, notebook grafik Anda sekarang harus dimigrasikan ke instance notebook Neptunus baru yang menggunakan lingkungan Amazon Linux 2 dan 4. JupyterLab Mereka akan muncul untuk akses dan manajemen di halaman Neptunus di Konsol Manajemen AWS, dan Anda sekarang dapat melanjutkan pekerjaan Anda dari tempat Anda tinggalkan dengan memilih Open Jupyter atau Open. JupyterLab

Buat notebook Neptunus di instans Amazon AI SageMaker

1. Buka konsol Amazon SageMaker AI di https://console.aws.amazon.com/sagemaker/.
2. Di panel navigasi, perluas Notebook, lalu pilih instance Notebook.
3. Pilih Buat instans notebook.
4. Dalam pengaturan instans Notebook, di bawah nama instance Notebook, beri buku catatan nama yang diawali dengan `aws-neptune-` (misalnya,`aws-neptune-my-test-notebook`).
5. Di bawah Pengenal platform, pilih Amazon Linux 2, JupyterLab 4.
6. Pilih Konfigurasi tambahan.
7. Di bawah Konfigurasi Siklus Hidup, pilih Buat konfigurasi siklus hidup baru.
8. Di Konfigurasi, di bawah Nama masukkan nama instance notebook dari langkah 4.
9. Di Skrip, di bawah Mulai buku catatan, ganti skrip yang ada dengan ini:

```
#!/bin/bash

sudo -u ec2-user -i <<'EOF'

echo "export GRAPH_NOTEBOOK_AUTH_MODE=IAM" >> ~/.bashrc
```

```
echo "export GRAPH_NOTEBOOK_SSL=True" >> ~/.bashrc
echo "export GRAPH_NOTEBOOK_SERVICE=neptune-db for Neptune, or neptune-graph for
 Neptune Analytics" >> ~/.bashrc
echo "export GRAPH_NOTEBOOK_HOST=(Neptune Analytics graph endpoint, public or
 private)" >> ~/.bashrc
echo "export GRAPH_NOTEBOOK_PORT=8182" >> ~/.bashrc
echo "export NEPTUNE_LOAD_FROM_S3_ROLE_ARN=" >> ~/.bashrc
echo "export AWS_REGION=(AWS region)" >> ~/.bashrc

aws s3 cp s3://aws-neptune-notebook-(AWS region)/graph_notebook.tar.gz /tmp/
graph_notebook.tar.gz
rm -rf /tmp/graph_notebook
tar -zxvf /tmp/graph_notebook.tar.gz -C /tmp
/tmp/graph_notebook/install_jl4x.sh

EOF
```

10. Pilih Buat konfigurasi.
11. Di Izin dan enkripsi, di bawah Peran IAM, pilih peran yang Anda buat di atas.
12. Di Jaringan, jika Anda menggunakan titik akhir grafik pribadi:
    a. Di bawah VPC, pilih VPC tempat grafik Neptunus Analytics berada.
    b. Di bawah Subnet, pilih subnet yang terkait dengan grafik Neptunus Analytics.
    c. Di bawah Grup Keamanan, pilih semua grup keamanan yang terkait dengan grafik Neptunus Analytics.
13. Pilih Buat instans notebook.
14. Setelah 5 atau 10 menit, ketika notebook baru Anda mencapai `Ready` status, pilih. Pilih Buka Jupyter atau Buka. JupyterLab

Menyiapkan notebook Neptunus secara manual

Anda juga dapat menggunakan paket AWS sumber terbuka yang tersedia untuk graph-notebook dan graph-explorer untuk menyiapkan lingkungan notebook Neptunus. Meskipun ada beberapa cara untuk menyiapkan notebook menggunakan paket sumber terbuka, pendekatan yang disarankan adalah:

- Siapkan grafik-notebook di komputer lokal Anda dan terowongan SSH Amazon EC2 yang menghubungkan mesin lokal Anda ke cluster Neptunus pada VPC yang sama dengan instans Amazon EC2.

- Siapkan graph-explorer pada instans Amazon EC2 dalam VPC Anda.

Untuk petunjuk terperinci tentang pengaturan paket graph-notebook dan graph-explorer open-source, lihat dokumentasi dan repositori resmi berikut: AWS GitHub

- https://docs.aws.amazon.com//neptune/latest/userguide/graph-notebooks.html#graph-notebooks-local
- https://docs.aws.amazon.com//neptune/latest/userguide/get-started-connect-ec2-same-vpc.html
- https://github.com/aws/graph-notebook/tree/main/additional-databases/neptune
- https://github.com/aws/graph-explorer/blob/main/additionaldocs/getting-started/README.md#amazon-ec2-setup

Konfigurasi keamanan

Saat menyiapkan lingkungan Anda, pastikan konfigurasi keamanan berikut diatur sesuai:

- Grup keamanan cluster Neptunus - Izinkan lalu lintas TCP yang masuk pada port 8182 dari grup keamanan instans Amazon EC2 Anda.
- Grup keamanan instans Amazon EC2 - Konfigurasikan aturan HTTPS masuk untuk mengaktifkan akses Graph Explorer.

Sebaiknya gunakan satu grup keamanan untuk cluster Neptunus dan instans Amazon EC2 Anda. Ini menyederhanakan konfigurasi dan mengurangi potensi masalah konektivitas. Namun, ingatlah untuk menambahkan aturan referensi diri untuk memungkinkan komunikasi antar sumber daya dalam grup.

Menggunakan sihir di notebook Amazon Neptunus

Workbench Neptune menyediakan sejumlah perintah yang disebut magic dalam notebook yang menghemat banyak waktu dan usaha. Perintah tersebut dibagi menjadi dua kategori: line magic dan cell magic.

Line magic adalah perintah yang didahului oleh tanda persen tunggal (%). Perintah ini hanya mengambil input baris, bukan input dari seluruh tubuh sel. Workbench Neptune menyediakan line magic berikut:

- %seed

- %load
- %load_ids
- %load_status
- %cancel_load
- %status
- %gremlin_status
- %opencypher_status, atau %oc_status
- `%stream_viewer`
- %sparql_status
- %graph_notebook_config
- %graph_notebook_host
- %graph_notebook_version
- %graph_notebook_service
- %graph_notebook_vis_options
- %statistik
- %ringkasan
- %reset_graph
- %get_graph
- %DegreeDistribusi

Cell magic didahului oleh dua tanda persen (`%%`) alih-alih satu, dan menggunakan konten sel sebagai input, meskipun perintah ini juga dapat mengambil konten baris sebagai input. Workbench Neptune menyediakan cell magic berikut:

- %%sparql
- %%gremlin
- %%opencypher, atau %%oc
- %%graph_notebook_config
- %%graph_notebook_vis_options

Terdapat juga dua magic, line magic dan cell magic, untuk bekerja dengan Machine learning Neptune:

- %neptune_ml
- %%neptune_ml

> **Note**
>
> Saat bekerja dengan sihir Neptunus, Anda biasanya bisa mendapatkan teks bantuan menggunakan parameter atau. `--help` `-h` Dengan sihir sel, tubuh tidak bisa kosong, jadi ketika mendapatkan bantuan, masukkan teks pengisi, bahkan satu karakter, ke dalam tubuh. Contoh:
>
> ```
> %%gremlin --help
>   x
> ```

Injeksi variabel dalam sihir sel atau garis

Variabel yang didefinisikan dalam buku catatan dapat direferensikan di dalam sihir sel atau baris apa pun di buku catatan menggunakan format:. `${VAR_NAME}`

Misalnya, Anda mendefinisikan variabel-variabel ini:

```
c = 'code'
my_edge_labels = '{"route":"dist"}'
```

Kemudian, kueri Gremlin ini dalam sihir sel:

```
%%gremlin -de $my_edge_labels
g.V().has('${c}','SAF').out('route').values('${c}')
```

Setara dengan ini:

```
%%gremlin -de {"route":"dist"}
g.V().has('code','SAF').out('route').values('code')
```

Argumen kueri yang berfungsi dengan semua bahasa kueri

Argumen kueri berikut bekerja dengan`%%gremlin`,`%%opencypher`, dan `%%sparql` sihir di meja kerja Neptunus:

Argumen kueri umum

- **`--store-to`**(atau **`-s`**) - Menentukan nama variabel di mana untuk menyimpan hasil query.
- **`--silent`**— Jika ada, tidak ada output yang ditampilkan setelah kueri selesai.

- **`--group-by`**(atau **`-g`**) - Menentukan properti yang digunakan untuk kelompok node (seperti `code` atau`T.region`). Simpul diwarnai berdasarkan grup yang ditugaskan.
- **`--ignore-groups`**— Jika ada, semua opsi pengelompokan diabaikan.
- **`--display-property`**(atau **`-d`**) - Menentukan properti yang nilainya harus ditampilkan untuk setiap simpul.

  Nilai default untuk setiap bahasa query adalah sebagai berikut:

  - Untuk Gremlin:. `T.label`
  - Untuk OpenCypher:. `~labels`
  - Untuk SPARQL:. `type`
- **`--edge-display-property`**(atau **`-t`**) - Menentukan properti yang nilainya harus ditampilkan untuk setiap tepi.

  Nilai default untuk setiap bahasa query adalah sebagai berikut:

  - Untuk Gremlin:. `T.label`
  - Untuk OpenCypher:. `~labels`
  - Untuk SPARQL:. `type`
- **`--tooltip-property`**(or **`-de`**) - Menentukan properti yang nilainya harus ditampilkan sebagai tooltip untuk setiap node.

  Nilai default untuk setiap bahasa query adalah sebagai berikut:

  - Untuk Gremlin:. `T.label`
  - Untuk OpenCypher:. `~labels`
  - Untuk SPARQL:. `type`
- **`--edge-tooltip-property`**(atau **`-te`**) - Menentukan properti yang nilainya harus ditampilkan sebagai tooltip untuk setiap tepi.

  Nilai default untuk setiap bahasa query adalah sebagai berikut:

  - Untuk Gremlin:. `T.label`
  - Untuk OpenCypher:. `~labels`

  - Untuk SPARQL:. `type`
- **`--label-max-length`** (atau **`-l`**) - Menentukan panjang karakter maksimum dari setiap label vertex. Default ke 10.
- **`--edge-label-max-length`** (atau **`-le`**) - Menentukan panjang karakter maksimum dari setiap label tepi. Default ke 10.

  Dalam kasus OpenCypher saja, ini adalah `--rel-label-max-length` atau sebaliknya. `-rel`
- **`--simulation-duration`**(atau **`-sd`**) — Menentukan durasi maksimum simulasi fisika visualisasi. Default ke 1500 ms.
- **`--stop-physics`**(atau **`-sp`**) — Menonaktifkan fisika visualisasi setelah simulasi awal stabil.

Nilai properti untuk argumen ini dapat terdiri dari kunci properti tunggal, atau string JSON yang dapat menentukan properti yang berbeda untuk setiap jenis label. Sebuah string JSON hanya dapat ditentukan menggunakan injeksi variabel.

Line magic `%seed`

Sihir `%seed` garis adalah cara mudah untuk menambahkan data ke titik akhir Neptunus Anda yang dapat Anda gunakan untuk menjelajahi dan bereksperimen dengan kueri Gremlin, OpenCypher, atau SPARQL. Ini menyediakan formulir di mana Anda dapat memilih model data yang ingin Anda jelajahi (grafik properti atau RDF) dan kemudian memilih di antara sejumlah kumpulan data sampel berbeda yang disediakan Neptunus.

Line magic `%load`

Line magic `%load` menghasilkan formulir yang dapat Anda gunakan untuk mengirimkan permintaan pemuatan massal ke Neptune (lihat Perintah Loader Neptune). Sumbernya harus berupa jalur Amazon S3 di wilayah yang sama dengan klaster Neptune.

Line magic `%load_ids`

Line magic `%load_ids` mengambil Id pemuatan yang telah dikirimkan ke titik akhir host notebook (lihat Parameter permintaan Neptune Loader Get-Status). Permintaan tersebut mengambil bentuk ini:

```
GET https://your-neptune-endpoint:port/loader
```

Line magic `%load_status`

Line magic `%load_status` mengambil status pemuatan pekerjaan pemuatan tertentu yang telah dikirimkan ke titik akhir host notebook, ditentukan oleh input baris (lihat Parameter permintaan Neptune Loader Get-Status). Permintaan tersebut mengambil bentuk ini:

```
GET https://your-neptune-endpoint:port/loader?loadId=loadId
```

Line magic terlihat seperti ini:

```
%load_status load id
```

Line magic `%cancel_load`

Line magic `%cancel_load` membatalkan pekerjaan pemuatan tertentu (lihat Pembatalan Pekerjaan Neptune Loader). Permintaan tersebut mengambil bentuk ini:

```
DELETE https://your-neptune-endpoint:port/loader?loadId=loadId
```

Line magic terlihat seperti ini:

```
%cancel_load load id
```

Line magic `%status`

Mengambil informasi status dari titik akhir host notebook (%graph_notebook_config menunjukkan titik akhir host).

Untuk host Neptunus DB, informasi status akan diambil dari titik akhir status kesehatan. Untuk host Neptune Analytics, status akan diambil melalui API. GetGraph Untuk informasi selengkapnya, lihat %get_graph.

Line magic `%gremlin_status`

Mengambil informasi status kueri Gremlin.

Sihir `%opencypher_status` garis (juga`%oc_status`)

Mengambil status kueri untuk kueri opencypher. Sihir baris ini mengambil argumen opsional berikut:

- **`--queryId`**or **`-q`**- Menentukan ID dari query berjalan tertentu untuk menampilkan status.
- **`--cancelQuery`**atau **`-c`**— Membatalkan kueri yang sedang berjalan. Tidak mengambil nilai.
- **`--silent-cancel`**atau **`-s`**— Jika `--silent` disetel ke `true` saat membatalkan kueri, kueri yang sedang berjalan dibatalkan dengan kode respons HTTP dari. `200` Jika tidak, kode respons HTTP akan menjadi`500`.
- **`--store-to`**- Menentukan nama variabel di mana untuk menyimpan hasil query.
- **`-w/--includeWaiting`**— Neptunus DB saja. Ketika disetel ke true dan parameter lainnya tidak ada, menyebabkan informasi status untuk kueri menunggu dikembalikan serta untuk menjalankan kueri. Parameter ini tidak mengambil nilai.
- **`--state`**— Neptunus Analytics saja. Menentukan apa subset dari status query untuk mengambil status.
- **`-m/--maxResults`**— Neptunus Analytics saja. Menetapkan batas atas pada set query dikembalikan cocok dengan nilai. `--state`
- **`--silent`**— Jika ada, tidak ada output yang ditampilkan setelah kueri selesai.

Line magic **`%stream_viewer`**

Sihir `%stream_viewer` garis menampilkan antarmuka yang memungkinkan untuk menjelajahi entri yang dicatat secara interaktif di aliran Neptunus, jika aliran diaktifkan di cluster Neptunus. Ia menerima argumen opsional berikut:

- **`language`**— Bahasa kueri dari data aliran: salah satu `gremlin` atau`sparql`. Defaultnya, jika Anda tidak memberikan argumen ini, adalah`gremlin`.
- **`--limit`**- Menentukan jumlah maksimum entri aliran untuk ditampilkan per halaman. Nilai default, jika Anda tidak memberikan argumen ini, adalah`10`.

> **Note**
>
> Sihir `%stream_viewer` garis sepenuhnya didukung hanya pada versi mesin 1.0.5.1 dan sebelumnya.

Line magic `%sparql_status`

Mengambil informasi status kueri SPARQL.

Line magic `%graph_notebook_config`

Line magic ini menampilkan objek JSON yang berisi konfigurasi yang notebook gunakan untuk berkomunikasi dengan Neptune. Konfigurasi ini meliputi:

- `host`: Titik akhir yang menghubungkan dan mengeluarkan perintah.
- `port`: Port yang digunakan saat mengeluarkan perintah ke Neptune. Nilai default-nya `8182`.
- `auth_mode`: Mode autentikasi yang digunakan saat mengeluarkan perintah ke Neptune. Harus `IAM` jika menghubungkan ke cluster yang memiliki otentikasi IAM diaktifkan, atau sebaliknya. `DEFAULT`
- `load_from_s3_arn`: Menentukan ARN Amazon S3 untuk digunakan magic `%load`. Jika nilai ini kosong, ARN harus ditentukan dalam perintah `%load`.
- `ssl`: Nilai Boolean yang menunjukkan apakah terhubung atau tidak ke Neptune menggunakan TLS. Nilai default-nya adalah `true`.
- `aws_region`: Wilayah tempat notebook ini di-deploy. Informasi ini digunakan untuk autentikasi IAM dan untuk permintaan `%load`.

Anda dapat mengubah konfigurasi ini dengan menyalin output `%graph_notebook_config` ke sel baru dan membuat perubahan di sana. Kemudian jika Anda menjalankan cell magic `%%graph_notebook_config` pada sel baru, konfigurasi akan berubah sewajarnya.

Line magic `%graph_notebook_host`

Menetapkan input baris sebagai host notebook.

Line magic `%graph_notebook_version`

Line magic `%graph_notebook_version` mengembalikan nomor rilis notebook workbench Neptune. Sebagai contoh, visualisasi grafik diperkenalkan dalam versi `1.27`.

Line magic `%graph_notebook_service`

Sihir `%graph_notebook_service` garis menetapkan input baris sebagai nama layanan yang digunakan untuk permintaan Neptunus.

Line magic `%graph_notebook_vis_options`

Line magic `%graph_notebook_vis_options` menampilkan pengaturan visualisasi saat ini yang digunakan notebook. Opsi ini dijelaskan dalam dokumentasi vis.js.

Anda dapat mengubah pengaturan ini dengan menyalin output ke sel baru, membuat perubahan yang Anda inginkan, dan kemudian menjalankan cell magic `%%graph_notebook_vis_options` di sel tersebut.

Untuk memulihkan pengaturan visualisasi ke nilai defaultnya, Anda dapat menjalankan line magic `%graph_notebook_vis_options` dengan parameter `reset`. Ini mengatur ulang semua pengaturan visualisasi:

```
%graph_notebook_vis_options reset
```

Line magic `%statistics`

Sihir `%statistics` garis digunakan untuk mengambil atau mengelola statistik mesin DFE (lihatMengelola statistik untuk Neptune DFE yang akan digunakan). Sihir ini juga dapat digunakan untuk mengambil ringkasan grafik.

Ini menerima parameter berikut:

- **`--language`**— Bahasa kueri dari titik akhir statistik: baik atau `propertygraph` (ataupg) ataurdf.

  Jika tidak disediakan, defaultnya adalahpropertygraph.
- **`--mode`**(atau **`-m`**) - Menentukan jenis permintaan atau tindakan untuk mengirimkan: salah satu daristatus,,disableAutoCompute,enableAutoCompute,refresh, deletedetailed, ataubasic).

  Jika tidak disediakan, defaultnya `--summary` adalah `status` kecuali ditentukan, dalam hal ini defaultnyabasic.
- **`--summary`**— Mengambil ringkasan grafik dari titik akhir ringkasan statistik dari bahasa yang dipilih.
- **`--silent`**— Jika ada, tidak ada output yang ditampilkan setelah kueri selesai.
- **`--store-to`**— Digunakan untuk menentukan variabel untuk menyimpan hasil query.

Line magic `%summary`

Sihir `%summary` garis digunakan untuk mengambil informasi ringkasan grafik. Ini tersedia mulai dengan versi mesin Neptunus. `1.2.1.0`

Ini menerima parameter berikut:

- **`--language`**— Bahasa kueri dari titik akhir statistik: baik atau `propertygraph` (ataupg) atau`rdf`.

  Jika tidak disediakan, defaultnya adalah`propertygraph`.
- **`--detailed`**— Mengaktifkan atau menonaktifkan tampilan bidang struktur dalam output.

  Jika tidak disediakan, defaultnya adalah mode tampilan `basic` ringkasan.
- **`--silent`**— Jika ada, tidak ada output yang ditampilkan setelah kueri selesai.
- **`--store-to`**— Digunakan untuk menentukan variabel untuk menyimpan hasil query.

Line magic `%reset_graph`

Sihir baris `%reset_graph` (atau`%_graph_reset`) mengeksekusi ResetGraphpanggilan terhadap titik akhir Neptunus Analytics. Perintah ini menerima input baris opsional berikut:

- -ns atau -- no-skip-snapshot - Jika ada, snapshot grafik akhir akan dibuat sebelum data grafik dihapus.
- --silent — Jika ada, tidak ada output yang ditampilkan setelah panggilan reset dikirimkan.
- --store-to — Digunakan untuk menentukan variabel untuk menyimpan respon. ResetGraph

Line magic `%get_graph`

Sihir `%get_graph` garis mengambil informasi tentang grafik melalui GetGraph API. Keajaiban ini secara fungsional identik dengan %status saat digunakan dengan Neptunus Analytics.

Line magic `%degreeDistribution`

Sihir `%degreeDistribution` garis menciptakan histogram interaktif yang memvisualisasikan distribusi derajat simpul dalam grafik. Histogram menampilkan jumlah simpul (sumbu y) yang memiliki derajat tertentu (sumbu x). Ini hanya tersedia dengan Neptune Analytics.

Ini menerima parameter berikut, yang juga dapat dipilih di menu tarik-turun:

- **`--traversalDirection`**— Menentukan tingkat mana yang akan dianalisis. Harus salah satu dari: `"both"` (default),`"inbound"`, atau`"outbound"`.
- **`--vertexLabels`**— Filter simpul dengan label yang disediakan (misalnya,`airport country`). Defaultnya adalah daftar kosong.
- **`--edgeLabels`**— Filter tepi dengan label yang disediakan (misalnya,`route`). Defaultnya adalah daftar kosong.

Cell magic **`%%sparql`**

Cell magic `%%sparql` mengeluarkan kueri SPARQL ke titik akhir Neptune. Perintah ini menerima input baris opsional berikut:

- **`-h`** atau **`--help`** — Mengembalikan teks bantuan tentang parameter ini.
- **`--path`** — Memberi prefiks pada jalur ke titik akhir SPARQL. Misalnya, jika Anda menentukan `--path "abc/def"` maka titik akhir yang dipanggil yaitu *`host`*`:`*`port`*`/abc/def`.
- **`--expand-all`** — Ini adalah petunjuk visualisasi kueri yang memberi tahu visualizer untuk menyertakan semua hasil `?s ?p ?o` dalam diagram grafik terlepas dari jenis mengikatnya.

  Secara default, visualisasi SPARQL hanya mencakup pola triple saat o? adalah `uri` atau bnode (node kosong). Semua jenis mengikat `?o` lainnya seperti string literal atau bilangan bulat diperlakukan sebagai properti dari node `?s` yang dapat dilihat menggunakan panel Detail di tab Grafik.

  Gunakan petunjuk kueri `--expand-all` ketika Anda mungkin ingin memasukkan nilai-nilai literal seperti vertex dalam visualisasi sebagai gantinya.

  Jangan gabungkan petunjuk visualisasi ini dengan parameter explain, karena kueri explain tidak divisualisasikan.
- **`--explain-type`** — Digunakan untuk menentukan mode explain yang akan digunakan (salah satu dari: `dynamic`, `static`, atau `details`).
- **`--explain-format`** — Digunakan untuk menentukan format respons untuk kueri explain (salah satu dari `text/csv` atau `text/html`).
- **`--store-to`**— Digunakan untuk menentukan variabel untuk menyimpan hasil query.

Contoh kueri `explain`:

```
%%sparql explain

SELECT * WHERE {?s ?p ?o} LIMIT 10
```

Contoh kueri visualisasi dengan parameter petunjuk visualisasi `--expand-all` (lihat Visualisasi SPARQL):

```
%%sparql --expand-all

SELECT * WHERE {?s ?p ?o} LIMIT 10
```

Cell magic **`%%gremlin`**

Sihir `%%gremlin` sel mengeluarkan kueri Gremlin ke titik akhir Neptunus menggunakan. WebSocket Perintah ini menerima input baris opsional untuk beralih ke mode /> `explain` Gremlin atau API `profile` Gremlin, dan input petunjuk visualisasi opsional terpisah untuk mengubah perilaku output visualisasi (lihat Visualisasi Gremlin).

Contoh kueri `explain`:

```
%%gremlin explain

g.V().limit(10)
```

Contoh kueri `profile`:

```
%%gremlin profile

g.V().limit(10)
```

Contoh kueri visualisasi dengan petunjuk kueri visualisasi:

```
%%gremlin -p v,outv

g.V().out().limit(10)
```

Parameter opsional untuk **`%%gremlin profile`** kueri

- **`--profile-chop`**- Menentukan panjang maksimum string hasil profil. Nilai default jika Anda tidak memberikan argumen ini adalah 250.

- **`--profile-serializer`**- Menentukan serializer untuk digunakan untuk hasil. Nilai yang diizinkan adalah salah satu nilai enum tipe MIME atau TinkerPop driver “Serializers” yang valid. Nilai default jika Anda tidak memberikan argumen ini adalah`application.json`.
- **`--profile-no-results`**— Hanya menampilkan jumlah hasil. Jika tidak digunakan, semua hasil kueri ditampilkan dalam laporan profil secara default.
- **`--profile-indexOps`**— Menampilkan laporan rinci dari semua operasi indeks.

Sihir **`%%opencypher`** sel (juga**`%%oc`**)

Sihir `%%opencypher` sel (yang juga memiliki `%%oc` bentuk disingkat), mengeluarkan kueri OpenCypher ke titik akhir Neptunus. Ia menerima argumen input baris opsional berikut:

- **`--mode`**— Mode kueri: salah satu `query` atau`bolt`. Nilai default jika Anda tidak memberikan argumen ini adalah`query`.
- **`--group-by`**or **`-g`**- Menentukan properti yang digunakan untuk kelompok node. Misalnya, `code, ~id`. Nilai default jika Anda tidak memberikan argumen ini adalah`~labels`.
- **`--ignore-groups`**— Jika ada, semua opsi pengelompokan diabaikan.
- **`--display-property`**or **`-d`**- Menentukan properti yang nilainya harus ditampilkan untuk setiap simpul. Nilai default jika Anda tidak memberikan argumen ini adalah`~labels`.
- **`--edge-display-property`**atau **`-de`**- Menentukan properti yang nilainya harus ditampilkan untuk setiap tepi. Nilai default jika Anda tidak memberikan argumen ini adalah`~labels`.
- **`--explain-type`**— Digunakan untuk menentukan mode jelaskan yang akan digunakan (salah satu dari: dinamis, statis, debug, atau detail).
- **`--label-max-length`**or **`-l`**- Menentukan jumlah maksimum karakter dari label simpul untuk ditampilkan. Nilai default jika Anda tidak memberikan argumen ini adalah`10`.
- **`--store-to`**atau **`-s`**- Menentukan nama variabel di mana untuk menyimpan hasil query.
- **`--plan-cache`**atau **`-pc`**- Menentukan modus rencana cache untuk digunakan. Nilai default-nya adalah `auto`.
- **`--query-timeout`**atau **`-qt`**- Menentukan batas waktu kueri maksimum dalam milidetik. Nilai default-nya adalah `1800000`.
- **`--query-parameters`**atau **`qp`**— Definisi parameter untuk diterapkan pada kueri. Opsi ini dapat menerima nama variabel tunggal, atau representasi string dari peta.

Contoh penggunaan **`--query-parameters`**

1. Tentukan peta parameter OpenCypher dalam satu sel notebook.

```
params = '''{
 "name":"john",
 "age": 20,
}'''
```

2. Masukkan parameter ke `--query-parameters` dalam sel lain dengan`%%oc`.

```
%%oc --query-parameters params

MATCH (n {name: $name, age: $age})
RETURN n
```

Cell magic **`%%graph_notebook_config`**

Cell magic `%%graph_notebook_config` menggunakan objek JSON yang berisi informasi konfigurasi untuk memodifikasi pengaturan yang notebook gunakan untuk berkomunikasi dengan Neptune, jika memungkinkan. Konfigurasi ini mengambil bentuk yang sama yang dikembalikan oleh line magic %graph_notebook_config.

Contoh:

```
%%graph_notebook_config
{
  "host": "my-new-cluster-endpoint.amazon.com",
  "port": 8182,
  "auth_mode": "DEFAULT",
  "load_from_s3_arn": "",
  "ssl": true,
  "aws_region": "us-east-1"
}
```

Cell magic **`%%graph_notebook_vis_options`**

Cell magic `%%graph_notebook_vis_options` memungkinkan Anda mengatur opsi visualisasi untuk notebook. Anda dapat menyalin pengaturan yang dikembalikan oleh line magic `%graph-`

`notebook-vis-options` ke dalam sel baru, membuat perubahan pada sel tersebut, dan menggunakan cell magic `%%graph_notebook_vis_options` untuk menetapkan nilai baru.

Opsi ini dijelaskan dalam dokumentasi vis.js.

Untuk memulihkan pengaturan visualisasi ke nilai defaultnya, Anda dapat menjalankan line magic `%graph_notebook_vis_options` dengan parameter `reset`. Ini mengatur ulang semua pengaturan visualisasi:

```
%graph_notebook_vis_options reset
```

Line magic `%neptune_ml`

Anda dapat menggunakan line magic `%neptune_ml` untuk menginisiasi dan mengelola berbagai operasi Neptune ML.

> **Note**
>
> Anda juga dapat menggunakan cell magic %%neptune_ml untuk menginisiasi dan mengelola beberapa operasi Neptune ML.

- **`%neptune_ml export start`** — Memulai pekerjaan ekspor baru.

  Parameter

  - **`--export-url`***`exporter-endpoint`*— (opsional) Titik akhir Amazon API Gateway tempat eksportir dapat dipanggil.
  - **`--export-iam`** — (opsional) Bendera yang menunjukkan bahwa permintaan ke url ekspor harus ditandatangani menggunakan SIGv4.
  - **`--export-no-ssl`** — (opsional) Bendera yang menunjukkan bahwa SSL tidak boleh digunakan saat menghubungkan ke exporter.
  - **`--wait`** — (opsional) Bendera yang menunjukkan bahwa operasi harus menunggu sampai ekspor selesai.
  - **`--wait-interval`***`interval-to-wait`*— (opsional) Mengatur waktu, dalam detik, antara pemeriksaan status ekspor (Default: 60).

  - **`--wait-timeout`***`timeout-seconds`*— (opsional) Mengatur waktu, dalam hitungan detik, untuk menunggu pekerjaan ekspor selesai sebelum mengembalikan status terbaru (Default: 3.600).
  - **`--store-to`***`location-to-store-result`*— (opsional) Variabel di mana untuk menyimpan hasil ekspor. Jika `--wait` ditentukan, status akhir akan disimpan di sana.
- **`%neptune_ml export status`** — Mengambil status pekerjaan ekspor.

  Parameter

  - **`--job-id`***`export job ID`*— ID pekerjaan ekspor untuk mengambil status.
  - **`--export-url`***`exporter-endpoint`*— (opsional) Titik akhir Amazon API Gateway tempat eksportir dapat dipanggil.
  - **`--export-iam`** — (opsional) Bendera yang menunjukkan bahwa permintaan ke url ekspor harus ditandatangani menggunakan SIGv4.
  - **`--export-no-ssl`** — (opsional) Bendera yang menunjukkan bahwa SSL tidak boleh digunakan saat menghubungkan ke exporter.
  - **`--wait`** — (opsional) Bendera yang menunjukkan bahwa operasi harus menunggu sampai ekspor selesai.
  - **`--wait-interval`***`interval-to-wait`*— (opsional) Mengatur waktu, dalam detik, antara pemeriksaan status ekspor (Default: 60).
  - **`--wait-timeout`***`timeout-seconds`*— (opsional) Mengatur waktu, dalam hitungan detik, untuk menunggu pekerjaan ekspor selesai sebelum mengembalikan status terbaru (Default: 3.600).
  - **`--store-to`***`location-to-store-result`*— (opsional) Variabel di mana untuk menyimpan hasil ekspor. Jika `--wait` ditentukan, status akhir akan disimpan di sana.
- **`%neptune_ml dataprocessing start`** — Memulai langkah pemrosesan data Neptune ML.

  Parameter

  - **`--job-id`***`ID for this job`*— (opsional) ID untuk ditetapkan ke pekerjaan ini.
  - **`--s3-input-uri`***`S3 URI`*— (opsional) URI S3 untuk menemukan input untuk pekerjaan pemrosesan data ini.
  - **`--config-file-name`***`file name`*— (opsional) Nama file konfigurasi untuk pekerjaan pemrosesan data ini.

- **`--store-to`***`location-to-store-result`*— (opsional) Variabel di mana untuk menyimpan hasil pemrosesan data.
- **`--instance-type`***`(instance type)`*— (opsional) Ukuran instans yang akan digunakan untuk pekerjaan pemrosesan data ini.
- **`--wait`** — (opsional) Bendera yang menunjukkan bahwa operasi harus menunggu sampai pemrosesan data selesai.
- **`--wait-interval`***`interval-to-wait`*— (opsional) Mengatur waktu, dalam detik, antara pemeriksaan status pemrosesan data (Default: 60).
- **`--wait-timeout`***`timeout-seconds`*— (opsional) Mengatur waktu, dalam hitungan detik, untuk menunggu pekerjaan pemrosesan data selesai sebelum mengembalikan status terbaru (Default: 3,600).

- **`%neptune_ml dataprocessing status`** — Mengambil status tugas pemrosesan data.

  Parameter

  - **`--job-id`***`ID of the job`*— ID pekerjaan untuk mengambil status.
  - **`--store-to`***`instance type`*— (opsional) Variabel untuk menyimpan hasil model-pelatihan.
  - **`--wait`** — (opsional) Bendera yang menunjukkan bahwa operasi harus menunggu sampai pelatihan model selesai.
  - **`--wait-interval`***`interval-to-wait`*— (opsional) Mengatur waktu, dalam detik, antara pemeriksaan status pelatihan model (Default: 60).
  - **`--wait-timeout`***`timeout-seconds`*— (opsional) Mengatur waktu, dalam hitungan detik, untuk menunggu pekerjaan pemrosesan data selesai sebelum mengembalikan status terbaru (Default: 3,600).

- **`%neptune_ml training start`** — Memulai proses pelatihan model Neptune ML.

  Parameter

  - **`--job-id`***`ID for this job`*— (opsional) ID untuk ditetapkan ke pekerjaan ini.
  - **`--data-processing-id`***`dataprocessing job ID`*— (opsional) ID pekerjaan pemrosesan data yang menciptakan artefak yang akan digunakan untuk pelatihan.
  - **`--s3-output-uri`***`S3 URI`*— (opsional) URI S3 untuk menyimpan output dari pekerjaan pelatihan model ini.
  - **`--instance-type`***`(instance type)`*— (opsional) Ukuran instans yang akan digunakan untuk pekerjaan pelatihan model ini.

- **`--store-to`***`location-to-store-result`*— (opsional) Variabel untuk menyimpan hasil model-pelatihan.
- **`--wait`** — (opsional) Bendera yang menunjukkan bahwa operasi harus menunggu sampai pelatihan model selesai.
- **`--wait-interval`***`interval-to-wait`*— (opsional) Mengatur waktu, dalam detik, antara pemeriksaan status pelatihan model (Default: 60).
- **`--wait-timeout`***`timeout-seconds`*— (opsional) Mengatur waktu, dalam hitungan detik, untuk menunggu pekerjaan pelatihan model selesai sebelum mengembalikan status terbaru (Default: 3.600).

- **`%neptune_ml training status`** — Mengambil status tugas pelatihan model Neptune ML.

  Parameter

  - **`--job-id`***`ID of the job`*— ID pekerjaan untuk mengambil status.
  - **`--store-to`***`instance type`*— (opsional) Variabel di mana untuk menyimpan hasil status.
  - **`--wait`** — (opsional) Bendera yang menunjukkan bahwa operasi harus menunggu sampai pelatihan model selesai.
  - **`--wait-interval`***`interval-to-wait`*— (opsional) Mengatur waktu, dalam detik, antara pemeriksaan status pelatihan model (Default: 60).
  - **`--wait-timeout`***`timeout-seconds`*— (opsional) Mengatur waktu, dalam hitungan detik, untuk menunggu pekerjaan pemrosesan data selesai sebelum mengembalikan status terbaru (Default: 3,600).

- **`%neptune_ml endpoint create`** — Menciptakan titik akhir kueri untuk model Neptune ML.

  Parameter

  - **`--job-id`***`ID for this job`*— (opsional) ID untuk ditetapkan ke pekerjaan ini.
  - **`--model-job-id`***`model-training job ID`*— (opsional) ID pekerjaan pelatihan model untuk membuat titik akhir kueri.
  - **`--instance-type`***`(instance type)`*— (opsional) Ukuran instance yang akan digunakan untuk titik akhir kueri..
  - **`--store-to`***`location-to-store-result`*— (opsional) Variabel di mana untuk menyimpan hasil pembuatan endpoint.
  - **`--wait`** — (opsional) Bendera yang menunjukkan bahwa operasi harus menunggu sampai pembuatan titik akhir selesai.

- **`--wait-interval`***`interval-to-wait`*— (opsional) Mengatur waktu, dalam detik, antara pemeriksaan status (Default: 60).
- **`--wait-timeout`***`timeout-seconds`*— (opsional) Mengatur waktu, dalam hitungan detik, untuk menunggu pekerjaan pembuatan titik akhir selesai sebelum mengembalikan status terbaru (Default: 3.600).

- **`%neptune_ml endpoint status`** — Mengambil status dari titik akhir kueri Neptune ML.

  Parameter

  - **`--job-id`***`endpoint creation ID`*— (opsional) ID pekerjaan pembuatan titik akhir untuk melaporkan statusnya.
  - **`--store-to`***`location-to-store-result`*— (opsional) Variabel di mana untuk menyimpan hasil status.
  - **`--wait`** — (opsional) Bendera yang menunjukkan bahwa operasi harus menunggu sampai pembuatan titik akhir selesai.
  - **`--wait-interval`***`interval-to-wait`*— (opsional) Mengatur waktu, dalam detik, antara pemeriksaan status (Default: 60).
  - **`--wait-timeout`***`timeout-seconds`*— (opsional) Mengatur waktu, dalam hitungan detik, untuk menunggu pekerjaan pembuatan titik akhir selesai sebelum mengembalikan status terbaru (Default: 3.600).

Cell magic **`%%neptune_ml`**

Cell magic `%%neptune_ml` mengabaikan input baris seperti `--job-id` atau `--export-url`. Sebaliknya, ini memungkinkan Anda memberikan input tersebut dan lainnya dalam tubuh sel.

Anda juga dapat menyimpan input tersebut di sel lain, ditugaskan ke variabel Jupyter, dan kemudian menyuntikkan mereka ke dalam tubuh sel menggunakan variabel itu. Dengan cara itu, Anda dapat menggunakan input tersebut berulang-ulang tanpa harus memasukkan kembali semua input tersebut setiap kalinya.

Ini hanya bekerja jika variabel suntik adalah satu-satunya konten dari sel. Anda tidak dapat menggunakan beberapa variabel dalam satu sel, atau kombinasi teks dan variabel.

Misalnya, cell magic `%%neptune_ml export start` dapat mengonsumsi dokumen JSON dalam tubuh sel yang berisi semua parameter yang dijelaskan dalam Parameter yang digunakan untuk mengontrol proses ekspor Neptunus.

Di notebook Neptune-ML-01-Introduction-to-Node-Classification-Gremlin, di bawah Mengkonfigurasi Fitur di bagian Ekspor data dan konfigurasi model, Anda dapat melihat bagaimana sel berikut memegang parameter ekspor dalam dokumen yang ditugaskan untuk variabel Jupyter bernama `export-params`:

```
export_params = {
  "command": "export-pg",
  "params": {
    "endpoint": neptune_ml.get_host(),
    "profile": "neptune_ml",
    "useIamAuth": neptune_ml.get_iam(),
    "cloneCluster": False
   },
  "outputS3Path": f'{s3_bucket_uri}/neptune-export',
  "additionalParams": {
    "neptune_ml": {
      "targets": [
        {
          "node": "movie",
          "property": "genre"
        }
      ],
      "features": [
        {
          "node": "movie",
          "property": "title",
          "type": "word2vec"
        },
        {
          "node": "user",
          "property": "age",
          "type": "bucket_numerical",
          "range" : [1, 100],
          "num_buckets": 10
        }
      ]
    }
  },
  "jobSize": "medium"}
```

Ketika Anda menjalankan sel ini, Jupyter menyimpan dokumen parameter dengan nama itu. Kemudian, Anda dapat menggunakan `${export_params}` untuk menyuntikkan dokumen JSON ke dalam tubuh `%%neptune_ml export start cell`, seperti ini:

```
%%neptune_ml export start --export-url {neptune_ml.get_export_service_host()} --export-
iam --wait --store-to export_results

${export_params}
```

Bentuk-bentuk yang tersedia dari cell magic **`%%neptune_ml`**

Cell magic `%%neptune_ml` dapat digunakan dalam bentuk berikut:

- **`%%neptune_ml export start`** — Memulai proses ekspor Neptune ML.
- **`%%neptune_ml dataprocessing start`** — Memulai tugas pemrosesan data Neptune ML.
- **`%%neptune_ml training start`** — Memulai tugas pelatihan model Neptune ML.
- **`%%neptune_ml endpoint create`** — Menciptakan titik akhir kueri Neptune ML untuk sebuah model.

Memvisualisasikan data grafik di notebook Amazon Neptunus

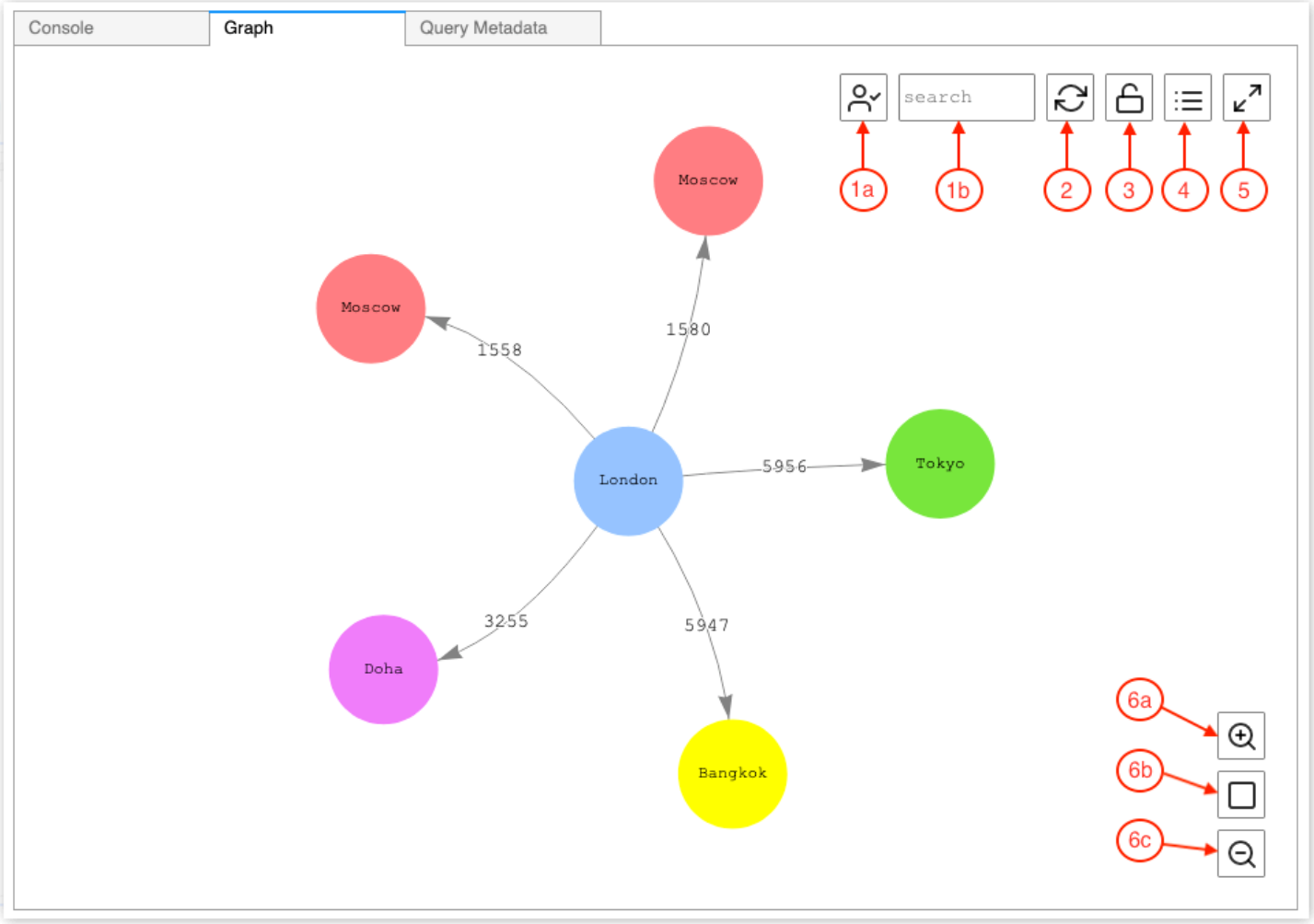
Dalam banyak kasus, workbench Neptune dapat membuat diagram visual dari hasil kueri Anda serta mengembalikannya dalam bentuk tabel. Visualisasi grafik tersedia di tab Grafik dalam hasil kueri setiap kali visualisasi dimungkinkan.

Selain kemampuan visualisasi bawaan yang dijelaskan di sini, Anda juga dapat menggunakan alat visualisasi yang lebih canggih dengan notebook grafik Neptunus.

> **Note**
>
> Untuk mendapatkan akses ke fungsi yang baru ditambahkan dan perbaikan di notebook yang Anda sudah gunakan, pertama berhentilah dan kemudian mulai ulang instans notebook Anda.

Menjelajahi antarmuka Tab Grafik di Amazon Neptunus

Diagram ini mengidentifikasi elemen antarmuka pengguna yang ada di tab Grafik:



1. Pencarian grafik
   a. UUID toggle: Beralih penyertaan nilai properti ID dalam pencarian grafik. Secara default, penyertaan ID diaktifkan. Jika dinonaktifkan, kecocokan pada properti ID, termasuk properti tepi yang mereferensikan simpul IDs, tidak menghasilkan penyorotan elemen.
   b. Cari bidang teks: Menyoroti semua nilai properti simpul dan tepi yang berisi string teks yang Anda tentukan di sini.
2. Reset grafik — Menjalankan kembali simulasi fisika grafik, dan mengatur zoom agar sesuai dengan grafik di jendela.
3. Beralih fisika grafik — Beralih menjalankan simulasi fisika grafik. Fisika diaktifkan secara default, membiarkan grafik berubah secara dinamis. Jika dinonaktifkan, simpul tetap terkunci pada posisinya saat simpul lain dipindahkan.
4. Tampilan detail - Ketika node atau tepi dipilih, ini menampilkan daftar kunci properti elemen dan nilai, jika tersedia dalam hasil kueri.

5. Tampilan layar penuh - Memperluas jendela tab grafik agar sesuai dengan layar. Mengklik lagi meminimalkan tab grafik.
6. Opsi zoom
   a. Memperbesar
   b. Zoom reset: Mengatur zoom agar sesuai dengan semua simpul di jendela tab grafik.
   c. Perkecil

Memvisualisasikan hasil kueri Gremlin

Workbench Neptune menciptakan visualisasi dari hasil kueri untuk setiap kueri Gremlin yang mengembalikan `path`. Untuk melihat visualisasinya, pilih tab Grafik di sebelah kanan tab Konsol di bawah kueri tersebut setelah Anda menjalankannya.

Anda dapat menggunakan petunjuk visualisasi kueri untuk mengontrol bagaimana visualizer membuat diagram output kueri. Petunjuk ini mengikuti cell magic `%%gremlin` dan didahului oleh nama parameter `--path-pattern` (atau bentuk pendeknya, `-p`):

```
%%gremlin -p comma-separated hints
```

Anda juga dapat menggunakan bendera `--group-by` (atau `-g`) untuk menentukan properti dari vertex untuk mengelompokkannya. Hal ini memungkinkan untuk menentukan warna atau ikon untuk grup vertex yang berbeda.

Nama-nama petunjuk mencerminkan langkah-langkah Gremlin yang umum digunakan ketika melakukan tranversing antara vertex, dan mereka berperilaku sesuai. Beberapa petunjuk dapat digunakan dalam kombinasi, dipisahkan dengan koma, tanpa spasi di antara keduanya. Petunjuk yang digunakan harus sesuai dengan langkah Gremlin yang sesuai dalam kueri yang divisualisasikan. Inilah contohnya:

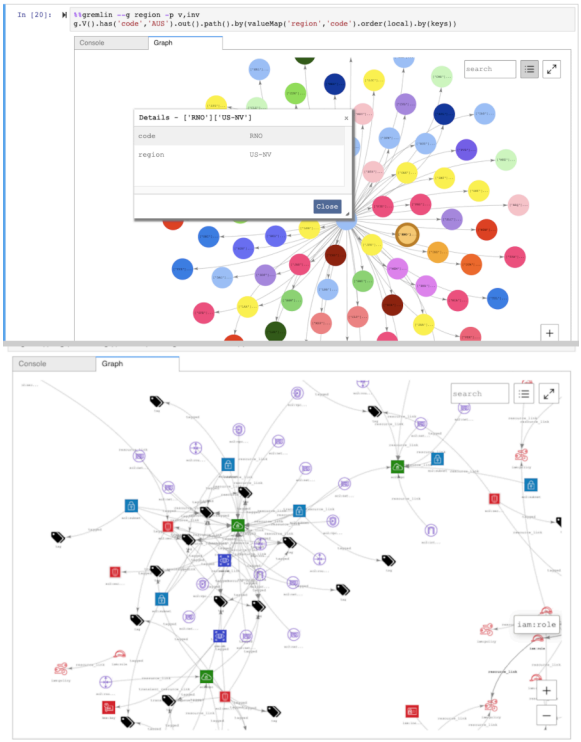
```
%%gremlin -p v,oute,inv
g.V().hasLabel('airport').outE().inV().path().by('code').by('dist').limit(5)
```

Petunjuk visualisasi yang tersedia adalah sebagai berikut:

```
v
inv
outv
```

```
e
ine
oute
```

Berikut ini beberapa contoh visualisasi grafik menggunakan grup:



Memvisualisasikan hasil kueri SPARQL

Workbench Neptune menciptakan visualisasi dari hasil kueri untuk setiap kueri SPARQL yang mengambil salah satu dari bentuk-bentuk ini:

- `SELECT ?subject ?predicate ?object`
- `SELECT ?s ?p ?o`

Untuk melihat visualisasinya, pilih tab Grafik di sebelah kanan tab Tabel di bawah kueri tersebut setelah Anda menjalankannya.

Secara default, visualisasi SPARQL hanya mencakup pola triple saat `o?` adalah `uri` atau `bnode` (node kosong). Semua jenis mengikat `?o` lainnya seperti string literal atau bilangan bulat diperlakukan sebagai properti dari node `?s` yang dapat dilihat menggunakan panel Detail di tab Grafik.

Namun, dalam banyak kasus, Anda mungkin ingin memasukkan nilai-nilai literal seperti vertex dalam visualisasi. Untuk melakukannya, gunakan petunjuk kueri `--expand-all` setelah cell magic `%%sparql`:

```
%%sparql --expand-all
```

Ini memberi tahu visualizer untuk menyertakan semua hasil `?s ?p ?o` dalam diagram grafik terlepas dari jenis mengikatnya.

Anda dapat melihat petunjuk ini digunakan di seluruh notebook `Air-Routes-SPARQL.ipynb` dan Anda dapat bereksperimen dengan menjalankan kueri dengan dan tanpa petunjuk tersebut untuk melihat perbedaan yang dia buat dalam visualisasi.

Mengakses notebook tutorial visualisasi di workbench Neptune

Dua notebook tutorial visualisasi yang hadir bersama workbench Neptune memberikan banyak contoh di Gremlin dan di SPARQL tentang bagaimana mengajukan kueri data grafik secara efektif dan memvisualisasikan hasilnya.

Bernavigasi ke notebook Visualisasi

1. Pada panel navigasi yang ada di sebelah kiri, pilih tombol Buka Notebook di sebelah kanan.
2. Setelah workbench Neptune terbuka, jalankan Jupyter, dan Anda akan melihat folder Neptune di tingkat atas. Pilih folder untuk membukanya.
3. Pada tingkat berikutnya adalah folder bernama 02-Visualisasi. Buka folder ini. Di dalamnya ada beberapa notebook yang memandu Anda melalui berbagai cara untuk mengajukan kueri data grafik Anda, di Gremlin dan di SPARQL, dan bagaimana memvisualisasikan hasil kuerinya:

   - Rute Udara-Gremlin
   - Rute udara-SPARQL
   - Blog Visualisasi Meja Kerja
   - EPL-Gremlin
   - EPL-SPARQL

   Pilih notebook untuk bereksperimen dengan kueri yang ada didalamnya.

Memecahkan masalah notebook Amazon Neptunus

Saat bekerja dengan notebook Neptunus JupyterLab di, Anda kadang-kadang dapat mengalami masalah yang terkait dengan perilaku browser, JupyterLab itu sendiri, atau integrasi Amazon AI. SageMaker Bagian berikut menjelaskan masalah yang paling umum dan cara mengatasinya.

Desinkronisasi status widget

Widget grafik-notebook terkadang dapat menjadi tidak sinkron antara UI dan backend kernel. JupyterLab Ketika ini terjadi, Anda mungkin melihat kesalahan tampilan widget atau pesan kesalahan seperti berikut:

```
Model class 'OutputModel' from module '@jupyter-widgets/output' is loaded but cannot be
 instantiated
```

Ini biasanya terjadi setelah periode idle yang lama, ketika beralih di antara tab notebook, setelah membersihkan cache browser, selama konektivitas jaringan yang tidak stabil, atau ketika membuka kembali notebook yang memiliki output widget yang belum disimpan.

Untuk mengatasinya, pilih Restart Kernel dan Hapus Semua Output dari menu Kernel, lalu jalankan kembali sel notebook Anda.

JupyterLab 4.x penundaan layar pemuatan awal

Saat membuka buku catatan untuk pertama kalinya setelah JupyterLab dimulai, Anda mungkin melihat layar pemuatan persisten dengan pesan: `Loading... The loading screen is taking a long time. Would you like to clear the workspace or keep waiting?`

Ini terutama mempengaruhi Firefox dan hanya terjadi pada notebook pertama yang terbuka dalam satu sesi. Anda dapat mengatasinya dengan memilih Clear Workspace, menyegarkan halaman browser, atau hanya membuka kembali buku catatan.

Jika Anda menggunakan Firefox dan mengalami ini berulang kali, coba hapus cookie dan data situs untuk domain SageMaker AI Anda di pengaturan Privasi & Keamanan Firefox, lalu segarkan JupyterLab halaman.

Masalah akses Graph Explorer

Setelah notebook dimulai ulang atau saat menggunakan kredensil yang di-cache, Anda mungkin mengalami kesulitan mengakses Graph Explorer. Ini dapat muncul sebagai kesalahan otentikasi, akses pesan yang ditolak, atau Graph Explorer gagal memuat seluruhnya.

Untuk mengatasinya, coba bersihkan cache dan cookie browser Anda. Anda juga dapat membuka JupyterLab atau Jupyter Classic terlebih dahulu untuk memaksa penyegaran kredenal, dan kemudian membuka Graph Explorer. Jika masalah berlanjut, coba gunakan jendela browser penyamaran atau pribadi.

Kiat umum

Banyak masalah notebook terkait browser. Untuk meminimalkan masalah, pertimbangkan hal berikut:

- Hapus cache browser Anda secara teratur saat bekerja dengan notebook Neptunus.
- Selalu gunakan sesi notebook terbaru daripada membuka kembali yang basi.
- Gunakan jendela browser penyamaran atau pribadi jika Anda mengalami masalah terus-menerus.
- Perbarui ekstensi buku catatan JupyterLab dan grafik-buku catatan Anda.

Menyiapkan Amazon Neptunus

Amazon Neptunus adalah layanan database grafik yang dikelola sepenuhnya yang membuatnya mudah untuk membangun dan menjalankan aplikasi yang bekerja dengan kumpulan data yang sangat terhubung. Untuk membantu Anda memulai Neptunus, bagian dokumentasi ini mencakup berbagai topik, mulai dari memilih jenis instans dan konfigurasi penyimpanan yang tepat hingga terhubung dengan aman ke cluster Neptunus, memuat data, dan memantau kinerja. Area utama yang tercakup dalam bagian ini adalah:

- Jenis Instance: Dokumentasi menguraikan berbagai jenis instans yang tersedia untuk Neptunus, beserta spesifikasi dan pertimbangannya untuk membantu Anda memilih opsi yang paling tepat untuk beban kerja Anda.
- Jenis Penyimpanan: Ini juga mencakup berbagai jenis penyimpanan yang didukung oleh Neptunus dan panduan tentang cara memilih konfigurasi penyimpanan yang tepat.
- Pembuatan Cluster: Dokumentasi memberikan step-by-step instruksi untuk membuat cluster Neptunus baru, termasuk cara mengonfigurasi pengaturan VPC yang diperlukan.
- Konektivitas dan Keamanan: Ini mencakup cara terhubung dengan aman ke cluster Neptunus Anda, mengatur akses VPC, dan mengonfigurasi langkah-langkah keamanan yang diperlukan untuk melindungi data Anda.
- Pemuatan Data: Dokumentasi menawarkan panduan tentang memuat data ke dalam cluster Neptunus Anda, termasuk opsi untuk pemuatan massal dan streaming data menggunakan berbagai alat dan. APIs
- Pemantauan dan Pemecahan Masalah: Ini juga mencakup informasi tentang pemantauan lingkungan Neptunus Anda, mengakses log, dan memecahkan masalah umum yang mungkin timbul selama operasi.

> **Note**
>
> Untuk arsitektur referensi database AWS grafik dan arsitektur penerapan referensi, Lihat Sumber Daya Amazon Neptunus. Sumber daya ini dapat membantu menginformasikan pilihan Anda tentang model data grafik dan bahasa kueri, serta mempercepat proses pengembangan Anda.

Memilih jenis instans untuk Amazon Neptunus

Amazon Neptunus menawarkan sejumlah ukuran instans dan keluarga yang berbeda. yang menawarkan kemampuan berbeda yang sesuai dengan beban kerja grafik yang berbeda. Bagian ini dimaksudkan untuk membantu Anda memilih jenis instans terbaik untuk kebutuhan Anda.

Untuk harga setiap jenis instans dalam keluarga ini, silakan lihat halaman harga Neptunus.

Ikhtisar alokasi sumber daya instance

Setiap jenis dan ukuran instans Amazon EC2 yang digunakan di Neptunus menawarkan jumlah komputasi (v) dan memori sistem yang ditentukan. CPUs Penyimpanan utama untuk Neptunus adalah eksternal dari instans DB dalam sebuah cluster, yang memungkinkan skala kapasitas komputasi dan penyimpanan secara independen satu sama lain.

Bagian ini berfokus pada bagaimana sumber daya komputasi dapat diskalakan, dan pada perbedaan antara masing-masing keluarga instance yang berbeda.

Di semua keluarga instance, sumber daya vCPU dialokasikan untuk mendukung dua (2) utas eksekusi kueri per vCPU. Dukungan ini ditentukan oleh ukuran instance. Saat menentukan ukuran yang tepat dari instans DB Neptunus tertentu, Anda perlu mempertimbangkan kemungkinan konkurensi aplikasi Anda dan latensi rata-rata kueri Anda. Anda dapat memperkirakan jumlah v yang CPUs dibutuhkan sebagai berikut, di mana latensi diukur sebagai latensi kueri rata-rata dalam detik dan konkurensi diukur sebagai jumlah target kueri per detik:

```
vCPUs = (latency x concurrency) / 2
```

> **Note**
>
> Kueri SPARQL, kueri OpenCypher, dan kueri baca Gremlin yang menggunakan mesin kueri DFE dapat, dalam keadaan tertentu, menggunakan lebih dari satu utas eksekusi per kueri. Saat awalnya mengukur cluster DB Anda, mulailah dengan asumsi bahwa setiap kueri akan menggunakan satu utas eksekusi per eksekusi dan tingkatkan jika Anda mengamati tekanan balik ke antrian kueri. Hal ini dapat diamati dengan menggunakan`/gremlin/status`,`/oc/status`, atau `/sparql/status` APIs, atau dapat juga diamati menggunakan `MainRequestsPendingRequestsQueue` CloudWatch metrik.

Memori sistem pada setiap instance dibagi menjadi dua alokasi utama: cache kumpulan buffer dan memori thread eksekusi kueri.

Sekitar dua pertiga dari memori yang tersedia dalam sebuah instance dialokasikan untuk cache buffer-pool. Cache buffer-pool digunakan untuk menyimpan komponen grafik yang paling baru digunakan untuk akses lebih cepat pada kueri yang berulang kali mengakses komponen tersebut. Instans dengan jumlah memori sistem yang lebih besar memiliki cache kumpulan buffer yang lebih besar yang dapat menyimpan lebih banyak grafik secara lokal. Pengguna dapat menyetel jumlah cache buffer-pool yang sesuai dengan memantau metrik hit dan miss cache buffer yang tersedia. CloudWatch

Anda mungkin ingin meningkatkan ukuran instans Anda jika tingkat hit cache turun di bawah 99,9% untuk jangka waktu yang konsisten. Ini menunjukkan bahwa kumpulan buffer tidak cukup besar, dan mesin harus mengambil data dari volume penyimpanan yang mendasarinya lebih sering daripada yang efisien.

Sepertiga sisanya dari memori sistem didistribusikan secara merata di seluruh thread eksekusi kueri, dengan beberapa memori yang tersisa untuk sistem operasi dan kolam dinamis kecil untuk thread untuk digunakan sesuai kebutuhan. Memori yang tersedia untuk setiap thread meningkat sedikit dari satu ukuran instance ke yang berikutnya hingga jenis `8xl` instance, di mana ukuran memori yang dialokasikan per thread mencapai maksimum.

Waktu untuk menambahkan lebih banyak memori thread adalah ketika Anda menemukan `OutOfMemoryException` (OOM). Pengecualian OOM terjadi ketika satu utas membutuhkan lebih dari memori maksimum yang dialokasikan untuknya (ini tidak sama dengan seluruh instance yang kehabisan memori).

`t3`dan jenis **`t4g`** contoh

The `t3` and `t4g` family of instance menawarkan opsi berbiaya rendah untuk memulai menggunakan database grafik dan juga untuk pengembangan dan pengujian awal. Instans ini memenuhi syarat untuk penawaran tingkat gratis Neptunus, yang memungkinkan pelanggan baru menggunakan Neptunus tanpa biaya untuk 750 jam instans pertama yang digunakan dalam akun AWS mandiri atau digulung di bawah Organisasi dengan Penagihan Konsolidasi (Akun Pembayar). AWS

`t4g`Instance `t3` dan hanya ditawarkan dalam konfigurasi ukuran sedang (`t3.mediumdant4g.medium`).

Mereka tidak dimaksudkan untuk digunakan dalam lingkungan produksi.

Karena instance ini memiliki sumber daya yang sangat terbatas, mereka tidak disarankan untuk menguji waktu eksekusi kueri atau kinerja database secara keseluruhan. Untuk menilai kinerja kueri, tingkatkan ke salah satu keluarga instance lainnya.

r4keluarga tipe instance

DEPRECATED — r4 Keluarga ini ditawarkan ketika Neptunus diluncurkan pada tahun 2018, tetapi sekarang jenis instans yang lebih baru menawarkan harga/kinerja yang jauh lebih baik. Pada versi mesin 1.1.0.0, Neptunus tidak lagi mendukung jenis instans. r4

r5keluarga tipe instance

r5Keluarga berisi jenis instance yang dioptimalkan untuk memori yang berfungsi dengan baik untuk sebagian besar kasus penggunaan grafik. r5Keluarga berisi tipe instance dari r5.large hinggar5.24xlarge. Mereka menskalakan secara linier dalam kinerja komputasi saat Anda meningkatkan ukuran. Misalnya, memori r5.xlarge (4 v CPUs dan 32GiB) memiliki dua kali v CPUs dan memori memori r5.large (2 v CPUs dan 16GiB), dan memori r5.2xlarge (8 v CPUs dan 64GiB) memiliki dua kali v dan memori memori. CPUs r5.xlarge Anda dapat mengharapkan kinerja kueri untuk diskalakan secara langsung dengan kapasitas komputasi hingga jenis r5.12xlarge instans.

Keluarga r5 instance memiliki arsitektur CPU Intel 2-soket. Tipe r5.12xlarge dan yang lebih kecil menggunakan soket tunggal dan memori sistem yang dimiliki oleh prosesor soket tunggal itu. r5.24xlargeJenis r5.16xlarge dan menggunakan soket dan memori yang tersedia. Karena ada beberapa overhead manajemen memori yang diperlukan antara dua prosesor fisik dalam arsitektur 2-soket, peningkatan kinerja yang ditingkatkan dari tipe a r5.12xlarge ke a r5.16xlarge atau r5.24xlarge instance tidak linier saat Anda meningkatkan skala pada ukuran yang lebih kecil.

r5dkeluarga tipe instance

Neptunus memiliki fitur lookup-cache yang dapat digunakan untuk meningkatkan kinerja query yang perlu mengambil dan mengembalikan sejumlah besar nilai properti dan literal. Fitur ini digunakan terutama oleh pelanggan dengan kueri yang perlu mengembalikan banyak atribut. Cache pencarian meningkatkan kinerja kueri ini dengan mengambil nilai atribut ini secara lokal daripada mencari masing-masing berulang kali di penyimpanan yang diindeks Neptunus.

Cache pencarian diimplementasikan menggunakan volume EBS NVMe -attached pada jenis instancer5d. Hal ini diaktifkan menggunakan kelompok parameter cluster. Karena data diambil

dari penyimpanan yang diindeks Neptunus, nilai properti dan literal RDF di-cache dalam volume ini. NVMe

Jika Anda tidak memerlukan fitur cache pencarian, gunakan jenis `r5` instance standar daripada`r5d`, untuk menghindari biaya yang lebih tinggi dari`r5d`.

`r5d`Keluarga memiliki tipe contoh dalam ukuran yang sama dengan `r5` keluarga, dari `r5d.large` hingga`r5d.24xlarge`.

`r6g`keluarga tipe instance

AWS telah mengembangkan prosesor berbasis ARM sendiri yang disebut Graviton, yang memberikan lebih baik price/performance daripada setara Intel dan AMD. `r6g`Keluarga menggunakan prosesor Graviton2. Dalam pengujian kami, prosesor Graviton2 menawarkan kinerja 10-20% lebih baik untuk kueri grafik gaya OLTP (dibatasi). Namun, kueri Olap-ish yang lebih besar mungkin sedikit kurang berkinerja dengan prosesor Graviton2 dibandingkan dengan prosesor Intel karena kinerja paging memori yang sedikit kurang berkinerja.

Penting juga untuk dicatat bahwa `r6g` keluarga memiliki arsitektur soket tunggal, yang berarti bahwa skala kinerja secara linier dengan kapasitas komputasi dari an `r6g.large` ke a `r6g.16xlarge` (tipe terbesar dalam keluarga).

`r6i`keluarga tipe instance

Instans Amazon R6i didukung oleh prosesor Intel Xeon Scalable generasi ke-3 (kode bernama Ice Lake) dan sangat cocok untuk beban kerja intensif memori. Sebagai aturan umum, mereka menawarkan kinerja harga komputasi hingga 15% lebih baik dan bandwidth memori per vCPU hingga 20% lebih tinggi daripada jenis instans R5 yang sebanding.

`x2g`keluarga tipe instance

Beberapa kasus penggunaan grafik melihat kinerja yang lebih baik ketika instance memiliki cache kumpulan buffer yang lebih besar. `x2g`Keluarga diluncurkan untuk mendukung kasus penggunaan tersebut dengan lebih baik. `x2g`Keluarga memiliki memory-to-vCPU rasio yang lebih besar daripada `r6g` keluarga `r5` atau keluarga. `x2g`Instans juga menggunakan prosesor Graviton2, dan memiliki banyak karakteristik kinerja yang sama dengan tipe `r6g` instance, serta cache buffer-pool yang lebih besar.

Jika Anda menggunakan `r5` atau jenis `r6g` instance dengan pemanfaatan CPU rendah dan tingkat kehilangan cache kumpulan buffer yang tinggi, coba gunakan keluarga sebagai `x2g` gantinya.

Dengan begitu, Anda akan mendapatkan memori tambahan yang Anda butuhkan tanpa membayar lebih banyak kapasitas CPU.

x2ieznkeluarga tipe instance

x2ieznKeluarga ini menyediakan instans yang dioptimalkan untuk memori yang didukung oleh prosesor Intel Xeon Scalable dengan kinerja frekuensi tinggi. Instans ini menawarkan memory-to-vCPU rasio tinggi (32 GiB per vCPU), membuatnya sangat cocok untuk beban kerja grafik intensif memori yang mendapat manfaat dari kinerja ulir tunggal yang tinggi.

Fitur utama termasuk hingga 4,5 frekuensi turbo GHz all-core dan ketersediaan dalam ukuran dari 2xlarge hingga 12xlarge.

x2iednkeluarga tipe instance

x2iednKeluarga ini menyediakan instance yang dioptimalkan untuk memori dengan penyimpanan SSD lokal NVMe . Instans ini menggabungkan kapasitas memori tinggi (32 GiB per vCPU) dengan penyimpanan lokal yang cepat, menjadikannya ideal untuk beban kerja grafik yang mendapat manfaat dari cache dalam memori besar dan cache disk lokal berkinerja tinggi.

Didukung oleh prosesor Intel Xeon Scalable generasi ke-3, instans ini tersedia dalam ukuran dari xlarge hingga 32xlarge dan dioptimalkan untuk database grafik skala besar yang membutuhkan kinerja memori dan penyimpanan.

r8gkeluarga tipe instance

r8gKeluarga berisi jenis instans yang dioptimalkan untuk memori yang didukung oleh prosesor AWS Graviton4. Instans ini menawarkan peningkatan kinerja yang signifikan dibandingkan generasi sebelumnya, membuatnya sangat cocok untuk beban kerja grafik intensif memori. Instans r8g memberikan kinerja sekitar 15-20% lebih baik untuk kueri grafik dibandingkan dengan instans r7g.

r8gKeluarga menggunakan platform dual-socket. Jenis instans dari r8g.large hingga r8g.24xlarge berjalan pada satu soket, yang berarti bahwa kinerja menskalakan secara linier dengan kapasitas komputasi di seluruh rentang tersebut. r8g.48xlargePenggunaan kedua soket dan merupakan jenis instans terbesar dalam keluarga; seperti keluarga dual-socket lainnya, peningkatan kinerja saat penskalaan dari r8g.24xlarge ke r8g.48xlarge mungkin tidak linier sempurna karena overhead manajemen memori cross-socket.

Fitur utama r8g keluarga meliputi:

- Didukung oleh prosesor berbasis AWS Graviton4 ARM
- Bandwidth memori per vCPU lebih tinggi dibandingkan generasi sebelumnya
- price/performance Rasio yang sangat baik untuk kueri grafik gaya OLTP (dibatasi) dan beban kerja analitik gaya OLAP
- Peningkatan kemampuan manajemen memori yang menguntungkan traversal grafik kompleks

`r8g`Keluarga ini ideal untuk beban kerja produksi yang membutuhkan kapasitas memori tinggi dan kinerja yang konsisten. Mereka sangat efektif untuk aplikasi dengan persyaratan konkurensi kueri tinggi.

`r7g`keluarga tipe instance

`r7g`Keluarga ini menggunakan prosesor AWS Graviton3, yang memberikan hasil yang lebih baik price/performance daripada instance berbasis Graviton2 sebelumnya. Dalam pengujian, prosesor Graviton3 menawarkan kinerja 25-30% lebih baik untuk kueri grafik gaya OLTP dibandingkan dengan instance r6g.

Seperti `r6g` keluarga, `r7g` keluarga memiliki arsitektur soket tunggal, yang berarti bahwa skala kinerja secara linier dengan kapasitas komputasi dari an `r7g.large` ke a `r7g.16xlarge` (tipe terbesar dalam keluarga).

Fitur utama `r7g` keluarga meliputi:

- Didukung oleh prosesor berbasis AWS Graviton3 ARM
- Peningkatan kinerja paging memori dibandingkan dengan r6g, menguntungkan beban kerja OLTP dan OLAP
- Peningkatan efisiensi cache buffer-pool
- Latensi yang lebih rendah untuk operasi intensif memori

`r7g`Keluarga ini sangat cocok untuk lingkungan produksi dengan pola kueri yang bervariasi dan sangat efektif untuk beban kerja yang mendapat manfaat dari peningkatan bandwidth memori.

`r7i`keluarga tipe instance

`r7i`Keluarga ini didukung oleh prosesor Intel Xeon Scalable generasi ke-4 (kode bernama Sapphire Rapids) dan menawarkan peningkatan signifikan dibandingkan instans r6i. Instans ini memberikan

komputasi sekitar 15% lebih baik price/performance dan bandwidth memori hingga 20% lebih tinggi per vCPU daripada jenis instans r6i yang sebanding.

Keluarga `r7i` instance memiliki arsitektur CPU Intel 2-soket, mirip dengan `r5` keluarga. Tipe `r7i.12xlarge` dan yang lebih kecil menggunakan soket tunggal dan memori sistem yang dimiliki oleh prosesor soket tunggal itu. `r7i.24xlarge`Jenis `r7i.16xlarge` dan menggunakan soket dan memori yang tersedia. Karena ada beberapa overhead manajemen memori yang diperlukan antara dua prosesor fisik dalam arsitektur 2-soket, peningkatan kinerja yang ditingkatkan dari tipe a `r7i.12xlarge` ke a `r7i.16xlarge` atau `r7i.24xlarge` instance tidak linier saat Anda meningkatkan skala pada ukuran yang lebih kecil.

Fitur utama `r7i` keluarga meliputi:

- Didukung oleh prosesor Intel Xeon Scalable generasi ke-4
- Skala kinerja linier dengan kapasitas komputasi hingga r7i.12xlarge
- Peningkatan manajemen memori antara prosesor fisik dalam arsitektur 2-soket
- Peningkatan kinerja untuk operasi grafik intensif memori

Untuk semua keluarga instance ini, Anda dapat memperkirakan jumlah v yang CPUs dibutuhkan menggunakan rumus yang sama yang disebutkan sebelumnya:

```
vCPUs = (latency x concurrency) / 2
```

Dimana latensi diukur sebagai latensi kueri rata-rata dalam hitungan detik dan konkurensi diukur sebagai jumlah target kueri per detik.

`serverless`jenis contoh

Fitur Neptunus Tanpa Server dapat menskalakan ukuran instans secara dinamis berdasarkan kebutuhan sumber daya beban kerja. Alih-alih menghitung berapa banyak v CPUs yang dibutuhkan untuk aplikasi Anda, Neptune Serverless memungkinkan Anda menetapkan batas bawah dan atas pada kapasitas komputasi (diukur dalam Unit Kapasitas Neptunus) untuk instance di cluster DB Anda. Beban kerja dengan pemanfaatan yang bervariasi dapat dioptimalkan biaya dengan menggunakan instans tanpa server daripada instance yang diprovsionasi.

Anda dapat menyiapkan instans yang disediakan dan tanpa server di cluster DB yang sama untuk mencapai konfigurasi kinerja biaya yang optimal.

Memilih jenis penyimpanan untuk Amazon Neptunus

Neptunus menawarkan dua jenis penyimpanan dengan model harga yang berbeda:

- Penyimpanan standar — Penyimpanan standar menyediakan penyimpanan database yang hemat biaya untuk aplikasi dengan penggunaan sedang hingga rendah I/O .
- I/O — Penyimpanan yang dioptimalkan — Dengan I/O–Optimized storage, available from engine version 1.3.0.0, you pay only for the storage and instances you are using. The storage costs are higher than for standard storage, and also the instance costs are higher than for standard instances. You pay nothing for the I/O that you use. If your I/O penggunaan yang tinggi, IOPs penyimpanan yang disediakan dapat mengurangi biaya secara signifikan.

  I/O–Optimized storage is designed to meet the needs of I/O–intensive graph workloads at a predictable cost. You can only switch between I/O—Jenis penyimpanan yang dioptimalkan dan Standar sekali per 30 hari.

  Untuk informasi harga tentang penyimpanan I/O—Optimized, lihat halaman harga Neptunus. Bagian berikut menjelaskan cara mengatur penyimpanan I/O—Optimized untuk cluster DB Neptunus.

Memilih penyimpanan I/O yang dioptimalkan untuk cluster DB Neptunus

Secara default, cluster DB Neptunus menggunakan penyimpanan standar.Anda dapat mengaktifkan penyimpanan I/O — Dioptimalkan pada cluster DB pada saat Anda membuatnya, seperti ini:

Berikut adalah contoh bagaimana Anda dapat mengaktifkan penyimpanan I/O—Optimized saat Anda membuat cluster menggunakan: AWS CLI

```
aws neptune create-db-cluster \
  --db-cluster-identifier (an ID for the cluster) \
  --engine neptune \
  --engine-version (the Neptune engine version) \
  --storage-type iopt1
```

Kemudian, setiap instance yang Anda buat secara otomatis mengaktifkan penyimpanan I/O—Optimized:

```
aws neptune create-db-instance \
```

```
  --db-cluster-identifier (the ID of the new cluster) \
  --db-instance-identifier (an ID for the new instance) \
  --engine neptune \
  --db-instance-class db.r5.large
```

Anda juga dapat memodifikasi cluster DB yang ada untuk mengaktifkan penyimpanan I/O—Optimized di dalamnya, seperti ini:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (the ID of a cluster without I/O-Optimized storage) \
  --storage-type iopt1 \
  --apply-immediately
```

Anda dapat memulihkan snapshot cadangan ke cluster DB dengan I/O—Penyimpanan yang dioptimalkan diaktifkan:

```
aws neptune restore-db-cluster-from-snapshot \
  --db-cluster-identifier (an ID for the restored cluster) \
  --snapshot-identifier (the ID of the snapshot to restore from) \
  --engine neptune \
  --engine-version (the Neptune engine version) \
  --storage-type iopt1
```

Anda dapat menentukan apakah sebuah cluster menggunakan I/O—Penyimpanan yang dioptimalkan menggunakan panggilan apa pun. `describe-` Jika penyimpanan I/O—Optimized diaktifkan, panggilan akan mengembalikan bidang tipe penyimpanan yang disetel ke. `iop1`

Membuat cluster Amazon Neptunus

Cara termudah untuk membuat cluster Amazon Neptunus DB baru adalah dengan menggunakan template CloudFormation yang membuat semua sumber daya yang diperlukan untuk Anda, tanpa harus melakukan semuanya dengan tangan. CloudFormation Template melakukan banyak penyiapan untuk Anda, termasuk membuat instance Amazon Elastic Compute Cloud (Amazon EC2):

Untuk meluncurkan cluster DB Neptunus baru menggunakan template CloudFormation

1. Buat pengguna IAM baru dengan izin yang Anda perlukan untuk bekerja dengan cluster DB Neptunus Anda, seperti yang dijelaskan di. Izin pengguna IAM

2. Siapkan prasyarat tambahan yang diperlukan untuk menggunakan CloudFormation template, seperti yang dijelaskan dalam. Prasyarat untuk mengatur Amazon Neptunus menggunakan AWS CloudFormation
3. Panggil CloudFormation tumpukan, seperti yang dijelaskan dalamMembuat cluster Amazon Neptunus menggunakan AWS CloudFormation.

Anda juga dapat membuat database global Neptunus yang mencakup Wilayah AWS beberapa, memungkinkan pembacaan global latensi rendah dan memberikan pemulihan cepat dalam kasus yang jarang terjadi di mana pemadaman mempengaruhi keseluruhan. Wilayah AWS

Untuk informasi tentang membuat klaster Amazon Neptunus secara manual menggunakan Konsol Manajemen AWS, lihat. Meluncurkan cluster DB Neptunus menggunakan Konsol Manajemen AWS

Anda juga dapat menggunakan CloudFormation template untuk membuat fungsi Lambda untuk digunakan dengan Neptunus (lihat). Menggunakan CloudFormation untuk Membuat Fungsi Lambda untuk Digunakan di Neptunus

Untuk informasi umum tentang mengelola cluster dan instance di Neptunus, lihat. Mengelola Basis Data Amazon Neptune Anda

Prasyarat untuk mengatur Amazon Neptunus menggunakan AWS CloudFormation

Sebelum Anda membuat cluster Amazon Neptunus menggunakan template, Anda harus CloudFormation memiliki yang berikut:

- Pasangan kunci Amazon EC2.
- Izin yang diperlukan untuk menggunakan CloudFormation.

Buat Pasangan Kunci Amazon EC2 yang akan digunakan untuk meluncurkan kluster Neptunus menggunakan CloudFormation

Untuk meluncurkan cluster DB Neptunus menggunakan CloudFormation template, Anda harus memiliki pasangan EC2key Amazon (dan file PEM terkait) yang tersedia di wilayah tempat Anda membuat tumpukan. CloudFormation

Jika Anda perlu membuat key pair, lihat Membuat Pasangan Kunci Menggunakan Amazon EC2 di Panduan Pengguna Amazon EC2, atau Membuat Pasangan Kunci Menggunakan Amazon EC2 di Panduan Pengguna Amazon EC2 untuk petunjuk.

Tambahkan kebijakan IAM untuk memberikan izin yang diperlukan untuk menggunakan templat CloudFormation

Pertama, Anda perlu mengatur pengguna IAM dengan izin yang diperlukan untuk bekerja dengan Neptunus, seperti yang dijelaskan dalam. Membuat pengguna IAM dengan izin untuk Neptunus

Maka Anda perlu menambahkan kebijakan AWS terkelola,`AWSCloudFormationReadOnlyAccess`, ke pengguna itu.

Terakhir, Anda perlu membuat kebijakan yang dikelola pelanggan berikut dan menambahkannya ke pengguna itu:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/*",
            "Condition": {
                "StringEquals": {
                    "iam:passedToService": "rds.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "rds.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics",
                "sns:ListSubscriptions",
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:*:111122223333:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:ListRetirableGrants",
                "kms:ListKeys",
                "kms:ListAliases",
                "kms:ListKeyPolicies"
            ],
            "Resource": "arn:aws:kms:*:111122223333:key/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
```

```
            "Resource": "arn:aws:cloudwatch:*:111122223333:service/*-*",
            "Condition": {
                "StringLike": {
                    "cloudwatch:namespace": "AWS/Neptune"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeVpcs",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute"
            ],
            "Resource": [
                "arn:aws:ec2:*:111122223333:vpc/*",
                "arn:aws:ec2:*:111122223333:subnet/*",
                "arn:aws:ec2:*:111122223333:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBCluster",
                "rds:CreateDBInstance",
                "rds:AddTagsToResource",
                "rds:ListTagsForResource",
                "rds:RemoveTagsFromResource",
                "rds:RemoveRoleFromDBCluster",
                "rds:ResetDBParameterGroup",
                "rds:CreateDBSubnetGroup",
                "rds:ModifyDBParameterGroup",
                "rds:DownloadDBLogFilePortion",
                "rds:CopyDBParameterGroup",
                "rds:AddRoleToDBCluster",
                "rds:ModifyDBInstance",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:DeleteDBInstance",
                "rds:CopyDBClusterParameterGroup",
                "rds:CreateDBParameterGroup",
```

```
                "rds:DescribeDBSecurityGroups",
                "rds:DeleteDBSubnetGroup",
                "rds:DescribeValidDBInstanceModifications",
                "rds:ModifyDBCluster",
                "rds:CreateDBClusterSnapshot",
                "rds:DeleteDBParameterGroup",
                "rds:CreateDBClusterParameterGroup",
                "rds:RemoveTagsFromResource",
                "rds:PromoteReadReplicaDBCluster",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribePendingMaintenanceActions",
                "rds:DescribeDBParameterGroups",
                "rds:FailoverDBCluster",
                "rds:DescribeDBInstances",
                "rds:DescribeDBParameters",
                "rds:DeleteDBCluster",
                "rds:ResetDBClusterParameterGroup",
                "rds:RestoreDBClusterToPointInTime",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:AddTagsToResource",
                "rds:DescribeDBClusterParameters",
                "rds:CopyDBClusterSnapshot",
                "rds:DescribeDBLogFiles",
                "rds:DeleteDBClusterSnapshot",
                "rds:ListTagsForResource",
                "rds:RebootDBInstance",
                "rds:DescribeDBClusterSnapshots",
                "rds:DeleteDBClusterParameterGroup",
                "rds:ApplyPendingMaintenanceAction",
                "rds:DescribeDBClusters",
                "rds:DescribeDBClusterParameterGroups",
                "rds:ModifyDBSubnetGroup"
            ],
            "Resource": [
                "arn:aws:rds:*:111122223333:cluster-snapshot:*",
                "arn:aws:rds:*:111122223333:cluster:*",
                "arn:aws:rds:*:111122223333:pg:*",
                "arn:aws:rds:*:111122223333:cluster-pg:*",
                "arn:aws:rds:*:111122223333:secgrp:*",
                "arn:aws:rds:*:111122223333:db:*",
                "arn:aws:rds:*:111122223333:subgrp:*"
            ],
            "Condition": {
```

```
                "StringEquals": {
                    "rds:DatabaseEngine": [
                        "graphdb",
                        "neptune"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:111122223333:log-group:*:log-stream:*",
                "arn:aws:logs:*:111122223333:log-group:*"
            ]
        }
    ]
}
```

> **Note**
>
> Izin berikut hanya diperlukan untuk menghapus tumpukan: `iam:DeleteRole`, `iam:RemoveRoleFromInstanceProfile`, `iam:DeleteRolePolicy`, `iam:DeleteInstanceProfile`, dan `ec2:DeleteVpcEndpoints`. Perhatikan juga bahwa `ec2:*Vpc` memberikan izin `ec2:DeleteVpc`.

Membuat cluster Amazon Neptunus menggunakan AWS CloudFormation

Anda dapat menggunakan CloudFormation template untuk menyiapkan Cluster DB Neptunus.

1. Untuk meluncurkan CloudFormation tumpukan di CloudFormation konsol, pilih salah satu tombol Launch Stack di tabel berikut.

Region	Lihat	Lihat di Designer	Luncurkan
US East (N. Virginia)	Lihat	Lihat di Desainer	
AS Timur (Ohio)	Lihat	Lihat di Desainer	
US West (N. California)	Lihat	Lihat di Desainer	
US West (Oregon)	Lihat	Lihat di Desainer	
Kanada (Pusat)	Lihat	Lihat di Desainer	
Kanada Barat (Calgary)	Lihat	Lihat di Desainer	
Amerika Selatan (Sao Paulo)	Lihat	Lihat di Desainer	
Europe (Stockholm)	Lihat	Lihat di Desainer	
Eropa (Irlandia)	Lihat	Lihat di Desainer	
Eropa (London)	Lihat	Lihat di Desainer	
Europe (Paris)	Lihat	Lihat di Desainer	

Region	Lihat	Lihat di Designer	Luncurkan
Eropa (Spanyol)	Lihat	Lihat di Desainer	
Eropa (Frankfurt)	Lihat	Lihat di Desainer	
Timur Tengah (Bahrain)	Lihat	Lihat di Desainer	
Timur Tengah (UAE)	Lihat	Lihat di Desainer	
Israel (Tel Aviv)	Lihat	Lihat di Desainer	
Africa (Cape Town)	Lihat	Lihat di Desainer	
Asia Pasifik (Hong Kong)	Lihat	Lihat di Desainer	
Asia Pacific (Tokyo)	Lihat	Lihat di Desainer	
Asia Pasifik (Seoul)	Lihat	Lihat di Desainer	
Asia Pacific (Singapore)	Lihat	Lihat di Desainer	
Asia Pacific (Sydney)	Lihat	Lihat di Desainer	
Asia Pasifik (Jakarta)	Lihat	Lihat di Desainer	
Asia Pacific (Melbourne)	Lihat	Lihat di Desainer	
Asia Pasifik (Malaysia)	Lihat	Lihat di Desainer	

Region	Lihat	Lihat di Designer	Luncurkan
Asia Pasifik (Mumbai)	Lihat	Lihat di Desainer	
Tiongkok (Beijing)	Lihat	Lihat di Desainer	
Tiongkok (Ningxia)	Lihat	Lihat di Desainer	
AWS GovCloud (AS-Barat)	Lihat	Lihat di Desainer	
AWS GovCloud (AS-Timur)	Lihat	Lihat di Desainer	

2. Pada halaman Pilih Templat, pilih Selanjutnya.
3. Pada halaman Specify Details, pilih key pair untuk file EC2SSHKeyPairName.

   Pasangan kunci ini diperlukan untuk mengakses instans EC2. Pastikan bahwa Anda memiliki file PEM untuk pasangan kunci yang Anda pilih.
4. Pilih Berikutnya.
5. Pada halaman Opsi, pilih Selanjutnya.
6. Pada halaman Review, pilih kotak centang pertama untuk mengetahui bahwa CloudFormation akan membuat sumber daya IAM. Pilih kotak centang kedua untuk mengetahui `CAPABILITY_AUTO_EXPAND` untuk tumpukan baru.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` secara eksplisit mengakui bahwa macro akan diperluas saat membuat tumpukan, tanpa review sebelumnya. Pengguna sering kali membuat perubahan yang ditetapkan dari templat yang diproses, sehingga perubahan yang dibuat oleh makro bisa direview tepat sebelum membuat tumpukan. Untuk informasi selengkapnya, lihat CloudFormation CreateStackAPI.

   Lalu pilih Buat.

> **Note**
>
> Anda juga dapat menggunakan CloudFormation template Anda untuk meningkatkan versi mesin cluster DB Anda.

Siapkan VPC Amazon di mana kluster DB Amazon Neptunus Anda berada

Sebuah klaster DB Amazon Neptune hanya dapat dibuat di Amazon Virtual Private Cloud (Amazon VPC). Titik akhirnya dapat diakses di dalam VPC itu, dan jika titik akhir publik Neptunus diaktifkan, mereka juga dapat diakses di luar VPC dan melalui internet.

Ada sejumlah cara berbeda untuk mengatur VPC, tergantung pada bagaimana Anda ingin mengakses cluster DB Anda.

Berikut adalah beberapa hal yang perlu diingat saat mengonfigurasi VPC tempat cluster DB Neptunus Anda berada:

- VPC Anda harus memiliki setidaknya dua subnet. Subnet ini harus berada dalam dua Availability Zones (AZs) yang berbeda. Dengan mendistribusikan instans cluster Anda di setidaknya dua, AZs Neptunus membantu memastikan bahwa selalu ada instance yang tersedia di cluster DB Anda bahkan jika terjadi kegagalan zona ketersediaan. Volume cluster untuk cluster DB Neptunus Anda selalu mencakup AZs tiga untuk menyediakan penyimpanan yang tahan lama dengan kemungkinan kehilangan data yang sangat rendah.
- Blok CIDR di setiap subnet harus cukup besar untuk memberikan alamat IP yang mungkin dibutuhkan Neptunus selama aktivitas pemeliharaan, failover, dan penskalaan.
- VPC harus memiliki grup subnet DB yang berisi subnet yang telah Anda buat. Neptunus memilih salah satu subnet dalam grup subnet dan alamat IP dalam subnet itu untuk dikaitkan dengan setiap instans DB di cluster DB. Instans DB kemudian terletak di AZ yang sama dengan subnet.
- VPC harus mengaktifkan DNS (baik nama host DNS dan resolusi DNS).
- VPC harus memiliki grup keamanan VPC untuk memungkinkan akses ke cluster DB Anda.
- Sewa di VPC Neptunus harus disetel ke Default.

Menambahkan subnet ke VPC tempat cluster DB Neptunus Anda berada

Subnet adalah serangkaian alamat IP di VPC. Anda dapat meluncurkan sumber daya seperti cluster DB Neptunus atau instans EC2 ke subnet tertentu. Saat Anda membuat subnet, Anda menentukan blok IPv4 CIDR untuk subnet, yang merupakan bagian dari blok CIDR VPC. Setiap subnet harus berada sepenuhnya dalam satu Availability Zone (AZ) dan tidak dapat menjangkau zona. Dengan meluncurkan instance di Availability Zone terpisah, Anda dapat melindungi aplikasi Anda dari kegagalan di salah satu zona. Lihat dokumentasi subnet VPC untuk informasi selengkapnya.

Cluster DB Neptunus membutuhkan setidaknya dua subnet VPC.

Untuk menambahkan subnet ke VPC

1. Masuk ke Konsol Manajemen AWS dan buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc/
2. Di panel navigasi, pilih Pengguna.
3. Di Dasbor VPC pilih Subnet, lalu pilih Buat subnet.
4. Pada halaman Create subnet, pilih VPC tempat Anda ingin membuat subnet.
5. Di bawah pengaturan Subnet, buat pilihan berikut:
    a. Masukkan nama untuk subnet baru di bawah nama Subnet.
    b. Pilih Availability Zone (AZ) untuk subnet, atau tinggalkan pilihan di No preference.
    c. Masukkan blok alamat IP subnet di bawah blok IPv4 CIDR.
    d. Tambahkan tag ke subnet jika perlu.
    e. Pilih
6. Jika Anda ingin membuat subnet lain secara bersamaan, pilih Tambahkan subnet baru.
7. Pilih Buat subnet untuk membuat subnet baru.

Mengkonfigurasi VPC di Amazon Neptunus

Buat grup subnet.

Untuk membuat grup subnet Neptunus

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/

2. Pilih Grup subnet, lalu pilih Buat Grup Subnet DB.
3. Masukkan nama dan deskripsi untuk grup subnet baru (deskripsi diperlukan).
4. Di bawah VPC, pilih VPC tempat Anda ingin grup subnet ini berada.
5. Di bawah zona Avalability, pilih AZ tempat Anda ingin grup subnet ini berada.
6. Di bawah Subnet, tambahkan satu atau lebih subnet di AZ ini ke grup subnet ini.
7. Pilih Buat untuk membuat grup subnet baru.

Buat grup keamanan menggunakan konsol VPC

Grup keamanan menyediakan akses ke cluster DB Neptunus Anda di VPC. Mereka bertindak sebagai firewall untuk cluster DB terkait, mengendalikan lalu lintas masuk dan keluar pada tingkat instans. Secara default, instans DB dibuat dengan firewall dan grup keamanan default yang mencegah akses ke sana. Untuk mengaktifkan akses, Anda harus memiliki grup keamanan VPC dengan aturan tambahan.

Prosedur berikut menunjukkan cara menambahkan aturan TCP kustom yang menentukan rentang port dan alamat IP untuk instans Amazon EC2 yang akan digunakan untuk mengakses kluster DB Neptunus Anda. Anda dapat menggunakan grup keamanan VPC yang ditetapkan ke instans EC2 daripada alamat IP-nya.

Untuk membuat grup keamanan VPC untuk Neptune di konsol

1. Masuk ke Konsol Manajemen AWS dan buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc/
2. Di sudut kanan atas konsol, pilih AWS wilayah tempat Anda ingin membuat grup keamanan VPC untuk Neptunus. Dalam daftar sumber daya VPC Amazon untuk wilayah itu, itu harus menunjukkan bahwa Anda memiliki setidaknya satu VPC dan beberapa subnet. Jika tidak, maka Anda tidak memiliki VPC default di Wilayah tersebut.
3. Di panel navigasi di bawah Keamanan, pilih Grup Keamanan.
4. Pilih Buat grup keamanan. Di jendela Buat grup keamanan, masukkan nama grup Keamanan, Deskripsi, dan pengidentifikasi VPC tempat cluster DB Neptunus Anda akan berada.
5. Tambahkan aturan masuk untuk grup keamanan instans Amazon EC2 yang ingin Anda sambungkan ke cluster DB Neptunus Anda:

    a. Di area Aturan masuk, pilih Tambahkan aturan.

b. Dalam daftar Jenis, biarkan TCP Kustom dipilih.
c. Dalam kotak Rentang port, masukkan 8182, nilai port default untuk Neptunus.
d. Di bawah Sumber, masukkan rentang alamat IP (nilai CIDR) dari mana Anda akan mengakses Neptunus, atau pilih nama grup keamanan yang ada.
e. Jika Anda perlu menambahkan lebih banyak alamat IP atau rentang port yang berbeda, pilih Tambah aturan lagi.

6. Di area Aturan keluar, Anda juga dapat menambahkan satu atau lebih aturan keluar jika perlu.
7. Setelah selesai, pilih Buat grup keamanan.

Anda dapat menggunakan grup keamanan VPC baru ini saat membuat cluster DB Neptunus baru.

Jika Anda menggunakan VPC default, grup subnet default yang mencakup semua subnet VPC sudah dibuat untuk Anda. Saat Anda memilih database Buat di konsol Neptunus, VPC default digunakan kecuali Anda menentukan yang berbeda.

Pastikan Anda memiliki dukungan DNS di VPC Anda

Sistem Nama Domain (DNS) adalah standar dimana nama yang digunakan di internet diubah ke alamat IP yang sesuai. Nama host DNS unik secara menjadi nama sebuah komputer dan terdiri dari nama host dan nama domain. Server DNS mengubah nama host DNS ke alamat IP yang sesuai.

Periksa untuk memastikan bahwa nama host DNS dan resolusi DNS keduanya diaktifkan di VPC Anda. Atribut jaringan VPC `enableDnsHostnames` dan `enableDnsSupport` harus diatur ke `true`. Untuk melihat dan memodifikasi atribut ini, buka konsol VPC di. https://console.aws.amazon.com/vpc/

Untuk informasi selengkapnya, lihat Menggunakan DNS dengan VPC Anda.

> **Note**
>
> Jika Anda menggunakan Route 53, konfirmasi bahwa konfigurasi Anda tidak menimpa atribut jaringan DNS di VPC Anda.

Menghubungkan ke cluster Amazon Neptunus

Setelah membuat cluster Neptunus, Anda dapat mengonfigurasi metode koneksi untuk mengaksesnya.

Menyiapkan `curl` atau mengawscurl untuk berkomunikasi dengan titik akhir Neptunus Anda

Memiliki alat baris perintah untuk mengirimkan kueri ke cluster DB Neptunus Anda sangat berguna, seperti yang diilustrasikan dalam banyak contoh dalam dokumentasi ini. Alat baris perintah curl adalah opsi yang sangat baik untuk berkomunikasi dengan titik akhir Neptunus ketika otentikasi IAM tidak diaktifkan. Versi yang dimulai dengan 7.75.0 mendukung `--aws-sigv4` opsi untuk menandatangani permintaan saat autentikasi IAM diaktifkan.

Untuk titik akhir di mana otentikasi IAM diaktifkan, Anda juga dapat menggunakan awscurl, yang menggunakan sintaks yang hampir persis sama `curl` tetapi mendukung permintaan penandatanganan seperti yang diperlukan untuk otentikasi IAM. Karena keamanan tambahan yang disediakan oleh autentikasi IAM, umumnya merupakan ide yang baik untuk mengaktifkannya.

Untuk informasi tentang cara menggunakan `curl` (atau`awscurl`), lihat halaman manual curl, dan buku Everything curl.

Untuk terhubung menggunakan HTTPS (yang dibutuhkan Neptunus)`curl`, perlu akses ke sertifikat yang sesuai. Selama `curl` dapat menemukan sertifikat yang sesuai, ia menangani koneksi HTTPS seperti koneksi HTTP, tanpa parameter tambahan. Hal yang sama berlaku untuk`awscurl`. Contoh dalam dokumentasi ini didasarkan pada skenario tersebut.

Untuk mempelajari cara mendapatkan sertifikat tersebut dan cara memformatnya dengan benar ke dalam penyimpanan sertifikat otoritas sertifikat (CA) yang `curl` dapat digunakan, lihat Verifikasi Sertifikat SSL dalam `curl` dokumentasi.

Anda kemudian dapat menentukan lokasi penyimpanan sertifikat CA ini menggunakan variabel lingkungan `CURL_CA_BUNDLE`. Pada Windows, `curl` secara otomatis mencarinya dalam sebuah file bernama `curl-ca-bundle.crt`. Ia pertama mencari dalam direktori yang sama dengan `curl.exe` dan kemudian di tempat lain di jalurnya. Untuk informasi lebih lanjut, lihat Verifikasi Sertifikat SSL.

Berbagai cara untuk terhubung ke cluster DB Neptunus

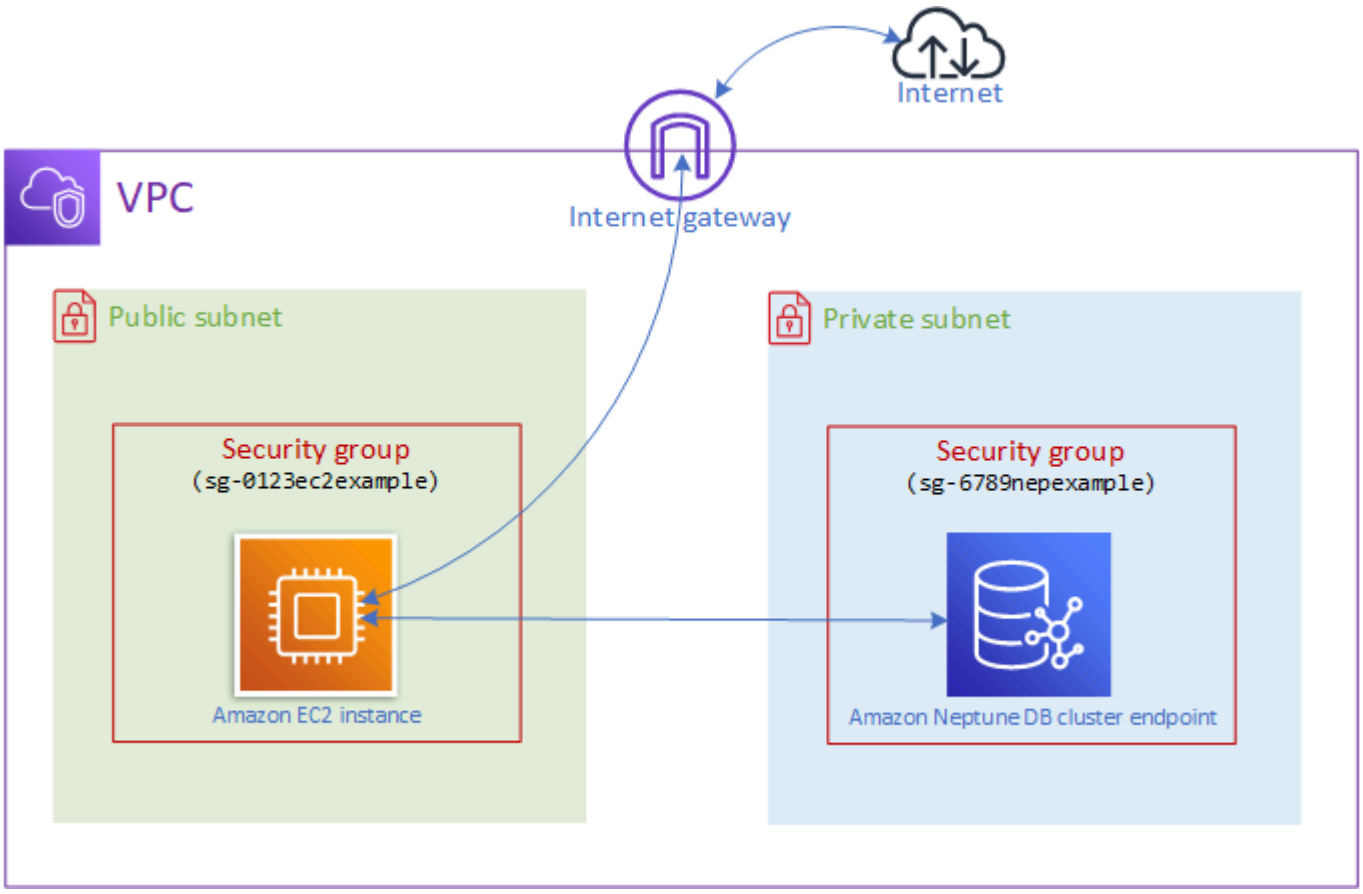
Sebuah klaster DB Amazon Neptune hanya dapat dibuat di Amazon Virtual Private Cloud (Amazon VPC). Titik akhirnya hanya dapat diakses dalam VPC itu kecuali Anda mengaktifkan dan mengatur titik akhir publik Neptunus untuk cluster DB.

Ada beberapa cara berbeda untuk mengatur akses ke cluster DB Neptunus Anda di VPC-nya:

- Menghubungkan dari instans Amazon EC2 di VPC yang sama
- Menghubungkan dari instans Amazon EC2 di VPC lain
- Menghubungkan dari jaringan pribadi
- Menghubungkan dari titik akhir publik

Menghubungkan instans Amazon EC2 ke cluster Amazon Neptunus di VPC yang sama

Salah satu cara paling umum untuk terhubung ke database Neptunus adalah dari instans Amazon EC2 di VPC yang sama dengan cluster DB Neptunus Anda. Misalnya, instans EC2 mungkin menjalankan server web yang terhubung dengan internet. Dalam hal ini, hanya instans EC2 yang memiliki akses ke cluster DB Neptunus, dan internet hanya memiliki akses ke instans EC2:



Untuk mengaktifkan konfigurasi ini, Anda harus menyiapkan grup keamanan VPC dan grup subnet yang tepat. Server web di-host di subnet publik, sehingga dapat menjangkau internet publik, dan instance cluster Neptunus Anda di-host di subnet pribadi untuk menjaganya tetap aman. Lihat Siapkan VPC Amazon di mana kluster DB Amazon Neptunus Anda berada.

Agar instans Amazon EC2 terhubung ke titik akhir Neptune Anda pada, misalnya, port 8182, Anda perlu menyiapkan grup keamanan untuk melakukannya. Jika instans Amazon EC2 Anda menggunakan grup keamanan bernama, misalnya, `ec2-sg1`, Anda perlu membuat grup keamanan Amazon EC2 lainnya (katakanlah `db-sg1`) yang memiliki aturan masuk untuk port `8182` dan memiliki `ec2-sg1` sebagai sumbernya. Kemudian, tambahkan `db-sg1` ke klaster Neptune Anda untuk mengizinkan koneksi.

Setelah membuat instans Amazon EC2, Anda dapat masuk ke dalamnya menggunakan SSH dan terhubung ke cluster DB Neptunus Anda. Untuk informasi tentang menghubungkan ke instans EC2 menggunakan SSH, lihat Connect to Linux Anda di Panduan Pengguna Amazon EC2.

Jika Anda menggunakan baris perintah Linux atau macOS untuk terhubung ke instans EC2, Anda dapat menempelkan perintah SSH dari SSHAccessitem di bagian Output tumpukan. CloudFormation Anda harus memiliki file PEM di direktori saat ini dan izin file PEM harus ditetapkan ke 400 (`chmod 400 keypair.pem`).

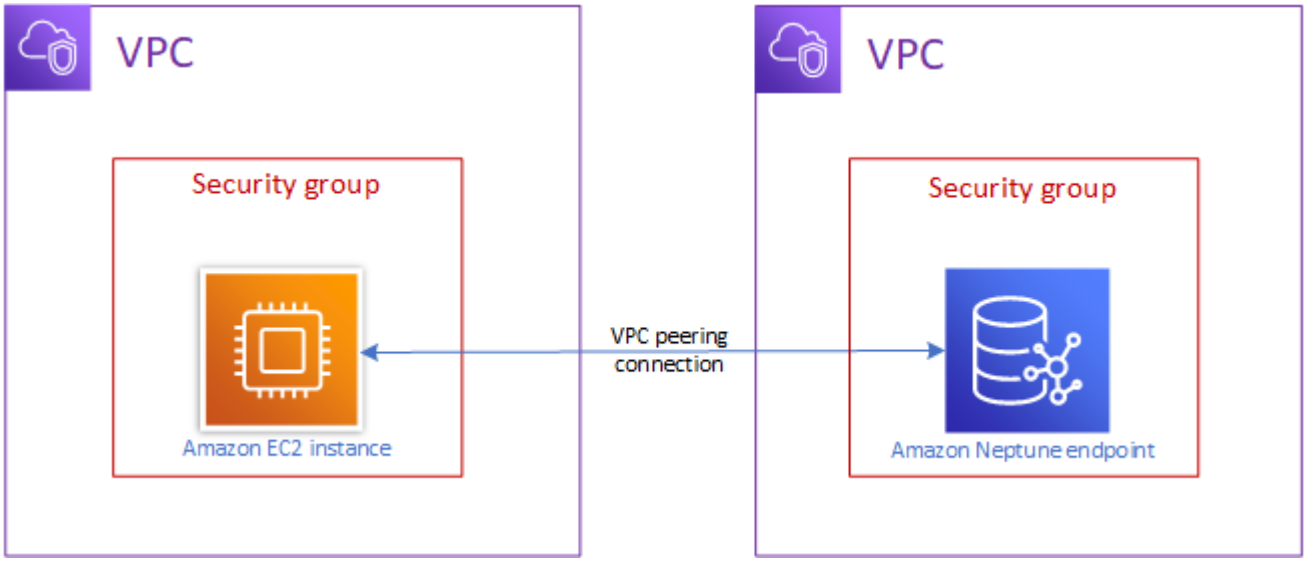
Untuk membuat VPC dengan subnet pribadi dan publik

1. Masuk ke Konsol Manajemen AWS dan buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc/
2. Di sudut kanan atas Konsol Manajemen AWS, pilih Wilayah untuk membuat VPC Anda.
3. Di Dasbor VPC, pilih Luncurkan Wisaya VPC.
4. Lengkapi area Pengaturan VPC di halaman Buat VPC:
    a. Di bawah Sumber daya untuk membuat, pilih VPC, subnet, dll.
    b. Biarkan tag nama default apa adanya, atau masukkan nama yang Anda pilih, atau hapus centang kotak centang Buat otomatis untuk menonaktifkan pembuatan tag nama.
    c. Biarkan nilai blok IPv4 CIDR di`10.0.0.0/16`.
    d. Biarkan nilai blok IPv6 CIDR di No IPv6 CIDR block.
    e. Tinggalkan Tenancy di Default.
    f. Biarkan jumlah Availability Zones (AZs) pada 2.
    g. Tinggalkan gateway NAT ($) di None, kecuali jika Anda membutuhkan satu atau lebih gateway NAT.
    h. Setel titik akhir VPC ke None, kecuali Anda akan menggunakan Amazon S3.
    i. Baik Aktifkan nama host DNS dan Aktifkan resolusi DNS harus diperiksa.
5. Pilih Buat VPC.

Menghubungkan instans Amazon EC2 ke cluster Amazon Neptunus di VPC yang berbeda

Cluster Amazon Neptune DB hanya dapat dibuat di Amazon Virtual Private Cloud (Amazon VPC), dan titik akhirnya dapat diakses dalam VPC itu, biasanya dari instans Amazon Elastic Compute Cloud (Amazon EC2) yang berjalan di VPC itu. Atau, dapat diakses menggunakan titik akhir publik. Untuk informasi selengkapnya tentang titik akhir publik, lihat titik akhir publik Neptunus.

Ketika cluster DB Anda berada di VPC yang berbeda dari instans EC2 yang Anda gunakan untuk mengaksesnya, Anda dapat menggunakan VPC peering untuk membuat koneksi:



Koneksi peering VPC adalah koneksi jaringan antara dua VPCs yang merutekan lalu lintas di antara mereka secara pribadi, sehingga contoh di salah satu VPC dapat berkomunikasi seolah-olah mereka berada dalam jaringan yang sama. Anda dapat membuat koneksi peering VPC antara VPCs di akun Anda, antara VPC di akun Anda AWS dan VPC di akun lain AWS , atau dengan VPC di Wilayah yang berbeda. AWS

AWS menggunakan infrastruktur VPC yang ada untuk membuat koneksi peering VPC. Ini bukan gateway atau koneksi AWS Site-to-Site VPN, dan tidak bergantung pada perangkat keras fisik yang terpisah. Ini tidak memiliki titik kegagalan tunggal untuk komunikasi dan tidak ada hambatan bandwidth.

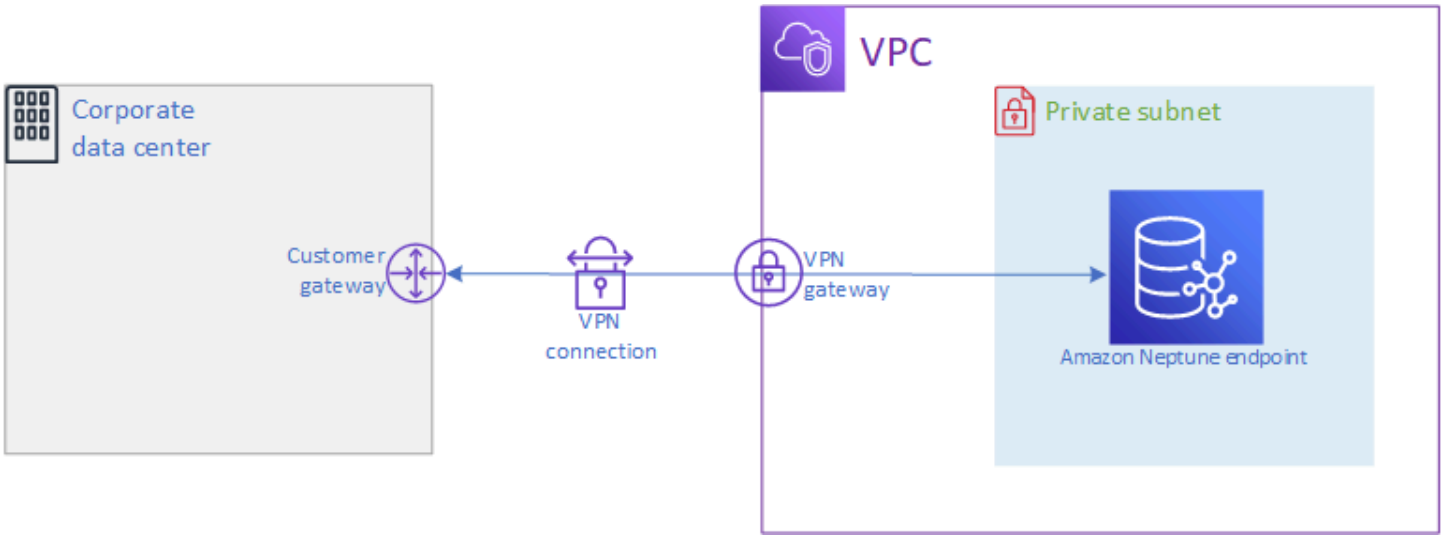
Lihat Panduan Peering VPC Amazon untuk informasi selengkapnya tentang cara menggunakan VPC peering.

Menghubungkan ke cluster Amazon Neptunus melalui jaringan pribadi

Anda dapat mengakses cluster DB Neptunus dari jaringan pribadi dengan dua cara berbeda:

- Menggunakan koneksi AWS Site-to-Site VPN.
- Menggunakan koneksi AWS Direct Connect.

Tautan di atas memiliki informasi tentang metode koneksi ini dan cara mengaturnya. Konfigurasi AWS Site-to-Site koneksi mungkin terlihat seperti ini:



Titik Akhir Publik Neptunus

Ikhtisar

Cluster Amazon Neptunus biasanya digunakan dalam VPC Anda dan hanya dapat diakses dari dalam VPC tersebut. Ini memerlukan konfigurasi aplikasi dan lingkungan pengembangan dalam VPC atau menggunakan layanan proxy untuk terhubung ke VPC, yang meningkatkan waktu dan biaya penyiapan.

Titik akhir publik menyederhanakan pengalaman ini dengan memungkinkan koneksi langsung ke Neptunus melalui internet, membuatnya lebih mudah untuk memulai dengan database grafik tanpa pengetahuan jaringan khusus.

Kapan menggunakan titik akhir publik

Pertimbangkan untuk menggunakan titik akhir publik dalam skenario berikut:

- Anda ingin menguji Neptunus dengan cepat dalam lingkungan pengembangan atau pengujian tanpa konfigurasi jaringan yang rumit
- Anda tidak memiliki pengetahuan AWS jaringan khusus
- Postur keamanan aplikasi Anda tidak memerlukan VPC pribadi
- Anda perlu terhubung ke Neptunus dari lingkungan pengembangan lokal Anda

Pertimbangan keamanan

Saat menggunakan titik akhir publik, ingatlah pertimbangan keamanan ini:

- Autentikasi IAM diperlukan untuk cluster dengan titik akhir publik diaktifkan.
- Akses ke database dikendalikan oleh grup keamanan yang digunakannya.
- Anda dapat membatasi alamat IP mana yang dapat terhubung ke cluster Anda.
- Anda dapat menggunakan kebijakan IAM untuk mengontrol siapa yang dapat membuat atau memodifikasi cluster dengan akses publik. Lihat: Membatasi pembuatan akses publik

Mengaktifkan titik akhir publik

Secara default, database Neptunus baru dibuat dengan titik akhir publik dinonaktifkan. Anda harus secara eksplisit mengaktifkan akses publik saat membuat atau memodifikasi klaster.

Titik akhir publik didukung dari rilis mesin Neptunus versi 1.4.6.x. Anda perlu memutakhirkan cluster yang ada ke setidaknya versi ini untuk menggunakan fitur ini.

Pengaturan titik akhir publik tersedia pada instance Neptunus dan bukan cluster Neptunus. Oleh karena itu, cluster Neptunus dapat eksis dengan beberapa contoh dengan titik akhir publik dan beberapa tidak. Namun, kami tidak menyarankan memiliki pengaturan seperti itu. Untuk informasi lebih lanjut tentang ini, lihat: Cara kerja titik akhir publik

Prasyarat

Pengaturan otentikasi IAM pada cluster Neptunus

Sebelum mengaktifkan titik akhir publik pada instans Neptunus, pastikan klaster Anda mendukung autentikasi IAM. Jika tidak, aktifkan menggunakan perintah berikut:

```
aws neptune modify-db-cluster \
```

```
  --region us-west-2 \
  --engine graphdb \
  --engine-version 1.4.6.x \
  --db-cluster-identifier neptune-public-endpoint \
  --enable-iam-database-authentication
```

Pengaturan jaringan

1. Pastikan VPC Anda memiliki subnet yang memungkinkan perutean publik (memiliki entri untuk gateway internet di tabel rute subnet). Jika Anda tidak memberikan `db-subnet-group-name` parameter saat membuat cluster, grup subnet default dipilih untuk pembuatan klaster.
2. Pastikan grup keamanan yang terpasang pada cluster memungkinkan lalu lintas masuk untuk rentang IP yang diizinkan dan port yang diizinkan. Misalnya, jika Anda ingin mengizinkan lalu lintas TCP dari semua IPs untuk terhubung ke instance Neptunus yang berjalan pada port 8182, aturan masuk harus memiliki:
   a. Jenis: Semua TCP
   b. Protokol: TCP
   c. Rentang port: 8182
   d. Blok CIDR: 0.0.0.0/0

> **Note**
>
> Meskipun Anda dapat mengatur rentang blok CIDR ke 0.0.0.0/0, kami sarankan untuk mengurangi ini ke rentang IP tertentu dari aplikasi klien Anda untuk postur keamanan yang lebih baik.

Membuat instance baru dengan titik akhir publik

Anda dapat membuat instance Neptunus baru dengan titik akhir publik menggunakan AWS Management Console, AWS CLI, atau SDK. AWS

Menggunakan AWS CLI:

```
aws neptune create-db-instance \
  --region us-west-2 \
  --engine graphdb \
  --engine-version 1.4.6.x \
```

```
  --db-cluster-identifier neptune-public-endpoint \
  --publicly-accessible
```

Memodifikasi instance yang ada untuk akses publik

Untuk memodifikasi instance Neptunus yang ada untuk mengaktifkan akses publik:

```
aws neptune modify-db-instance \
  --region us-west-2 \
  --engine graphdb \
  --engine-version 1.4.6.x \
  --db-instance-identifier neptune-public-endpoint \
  --publicly-accessible
```

> **Note**
>
> Akses publik diaktifkan pada tingkat instans, bukan tingkat cluster. Untuk memastikan klaster Anda selalu dapat diakses melalui titik akhir publik, semua instance di klaster harus mengaktifkan akses publik.

Menggunakan titik akhir publik

Untuk memeriksa apakah database Anda dapat dijangkau, periksa status menggunakan CLI API AWS : `NeptuneData`

```
aws neptunedata get-engine-status \
  --endpoint-url https://my-cluster-name.cluster-abcdefgh1234.us-
east-1.neptune.amazonaws.com:8182
```

Jika database dapat diakses, responsnya seperti:

```
{
    "status": "healthy",
    "startTime": "Sun Aug 10 06:54:15 UTC 2025",
    "dbEngineVersion": "1.4.6.0.R1",
    "role": "writer",
    "dfeQueryEngine": "viaQueryHint",
    "gremlin": {
        "version": "tinkerpop-3.7.1"
```

```
    },
    "sparql": {
        "version": "sparql-1.1"
    },
    "opencypher": {
        "version": "Neptune-9.0.20190305-1.0"
    },
    "labMode": {
        "ObjectIndex": "disabled",
        "ReadWriteConflictDetection": "enabled"
    },
    "features": {
        "SlowQueryLogs": "disabled",
        "InlineServerGeneratedEdgeId": "disabled",
        "ResultCache": {
            "status": "disabled"
        },
        "IAMAuthentication": "disabled",
        "Streams": "disabled",
        "AuditLog": "disabled"
    },
    "settings": {
        "StrictTimeoutValidation": "true",
        "clusterQueryTimeoutInMs": "120000",
        "SlowQueryLogsThreshold": "5000"
    }
}
```

Contoh cara menanyakan database

AWS CLI

```
aws neptunedata execute-open-cypher-query \
--open-cypher-query "MATCH (n) RETURN n LIMIT 10" \
--endpoint-url https://my-cluster-name.cluster-abcdefgh1234.us-
east-1.neptune.amazonaws.com:8182
```

Python

```
import boto3
import json
from botocore.config import Config
```

```
# Configuration - Replace with your actual Neptune cluster details
cluster_endpoint = "my-cluster-name.cluster-abcdefgh1234.my-
region.neptune.amazonaws.com"
port = 8182
region = "my-region"

# Configure Neptune client
# This disables retries and sets the client timeout to infinite
#     (relying on Neptune's query timeout)
endpoint_url = f"https://{cluster_endpoint}:{port}"
config = Config(
    region_name=region,
    retries={'max_attempts': 1},
    read_timeout=None
)

client = boto3.client("neptunedata", config=config, endpoint_url=endpoint_url)

cypher_query = "MATCH (n) RETURN n LIMIT 5"
try:
    response = client.execute_open_cypher_query(openCypherQuery=cypher_query)
    print("openCypher Results:")
    for item in response.get('results', []):
        print(f"  {item}")
except Exception as e:
    print(f"openCypher query failed: {e}")
```

JavaScript

```
import {
    NeptunedataClient,
    GetPropertygraphSummaryCommand
} from "@aws-sdk/client-neptunedata";
import { inspect } from "util";
import { NodeHttpHandler } from "@smithy/node-http-handler";

/**
 * Main execution function
 */
async function main() {
    // Configuration - Replace with your actual Neptune cluster details
    const clusterEndpoint = 'my-cluster-name.cluster-abcdefgh1234.my-
region.neptune.amazonaws.com';
```

```
    const port = 8182;
    const region = 'my-region';

    // Configure Neptune client
    // This disables retries and sets the client timeout to infinite
    //     (relying on Neptune's query timeout)
    const endpoint = `https://${clusterEndpoint}:${port}`;
    const clientConfig = {
        endpoint: endpoint,
        sslEnabled: true,
        region: region,
        maxAttempts: 1,  // do not retry
        requestHandler: new NodeHttpHandler({
            requestTimeout: 0  // no client timeout
        })
    };

    const client = new NeptunedataClient(clientConfig);
    try {
        try {
            const command = new GetPropertygraphSummaryCommand({ mode: "basic" });
            const response = await client.send(command);
            console.log("Graph Summary:", inspect(response.payload, { depth: null }));
        } catch (error) {
            console.log("Property graph summary failed:", error.message);
        }
    } catch (error) {
        console.error("Error in main execution:", error);
    }
}

// Run the main function
main().catch(console.error);
```

Go

```
package main
import (
    "context"
    "fmt"
    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/neptunedata"
```

```
    "os"
    "encoding/json"
    "net/http"
)

func main() {
    // Configuration - Replace with your actual Neptune cluster details
    clusterEndpoint := "my-cluster-name.cluster-abcdefgh1234.my-
region.neptune.amazonaws.com"
    port := 8182
    region := "my-region"

    // Configure Neptune client
    // Configure HTTP client with no timeout
    //    (relying on Neptune's query timeout)
    endpoint := fmt.Sprintf("https://%s:%d", clusterEndpoint, port)
    // Load AWS SDK configuration
    sdkConfig, _ := config.LoadDefaultConfig(
        context.TODO(),
        config.WithRegion(region),
        config.WithHTTPClient(&http.Client{Timeout: 0}),
    )

    // Create Neptune client with custom endpoint
    client := neptunedata.NewFromConfig(sdkConfig, func(o *neptunedata.Options) {
        o.BaseEndpoint = aws.String(endpoint)
        o.Retryer = aws.NopRetryer{} // Do not retry calls if they fail
    })

    gremlinQuery :=
 "g.addV('person').property('name','charlie').property(id,'charlie-1')"
    serializer := "application/vnd.gremlin-v1.0+json;types=false"

    gremlinInput := &neptunedata.ExecuteGremlinQueryInput{
        GremlinQuery: &gremlinQuery,
        Serializer:   &serializer,
    }
    gremlinResult, err := client.ExecuteGremlinQuery(context.TODO(), gremlinInput)
    if err != nil {
        fmt.Printf("Gremlin query failed: %v\n", err)
    } else {
        var resultMap map[string]interface{}
        err = gremlinResult.Result.UnmarshalSmithyDocument(&resultMap)
        if err != nil {
```

```
            fmt.Printf("Error unmarshaling Gremlin result: %v\n", err)
        } else {
            resultJSON, _ := json.MarshalIndent(resultMap, "", "  ")
            fmt.Printf("Gremlin Result: %s\n", string(resultJSON))
        }
    }
}
```

Bagaimana titik akhir publik bekerja

Ketika instance Neptunus dapat diakses publik:

- Titik akhir DNS-nya menyelesaikan ke alamat IP pribadi dari dalam VPC cluster DB.
- Ini menyelesaikan ke alamat IP publik dari luar VPC cluster.
- Akses dikendalikan oleh grup keamanan yang ditugaskan ke cluster.
- Hanya contoh yang dapat diakses publik yang dapat diakses melalui internet.

Perilaku titik akhir pembaca

- Jika semua instance pembaca dapat diakses publik, titik akhir pembaca akan selalu diselesaikan melalui internet publik.
- Jika hanya beberapa instance pembaca yang dapat diakses publik, titik akhir pembaca akan menyelesaikan secara publik hanya jika memilih instance yang dapat diakses publik untuk melayani kueri baca.

Perilaku titik akhir cluster

- Titik akhir cluster DB selalu menyelesaikan titik akhir instance penulis.
- Jika titik akhir publik diaktifkan pada instance penulis, titik akhir cluster akan dapat diakses publik, jika tidak maka tidak akan terjadi.

Perilaku setelah failover cluster

- Cluster Neptunus dapat memiliki instance pada pengaturan yang dapat diakses publik yang berbeda.

- Jika cluster memiliki penulis publik dan pembaca non-publik, posting failover cluster, penulis baru (pembaca sebelumnya) menjadi non-publik dan pembaca baru (penulis sebelumnya) menjadi publik.

Persyaratan konfigurasi jaringan

Agar titik akhir publik berfungsi dengan baik:

1. Instans Neptunus harus berada dalam subnet publik dalam VPC Anda.
2. Tabel rute yang terkait dengan subnet ini harus memiliki rute ke gateway internet untuk 0.0.0.0/0.
3. Grup keamanan harus mengizinkan akses dari alamat IP publik atau rentang CIDR yang ingin Anda berikan akses.

Membatasi pembuatan akses publik

Anda dapat menggunakan kebijakan IAM untuk membatasi siapa yang dapat membuat atau memodifikasi kluster Neptunus dengan akses publik. Contoh kebijakan berikut menyangkal pembuatan instance Neptunus dengan akses publik:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "rds:CreateDBInstance",
        "rds:ModifyDBInstance",
        "rds:RestoreDBInstanceFromDBSnapshot",
        "rds:RestoreDBInstanceToPointInTime"
      ],
      "Resource": "*",
      "Condition": {
        "Bool": {
          "rds:PubliclyAccessible": true
        }
      }
    }
```

```
  ]
}
```

Lebih lanjut tentang kunci kondisi `rds:PublicAccessEnabled` IAM: Referensi Otorisasi Layanan Amazon RDS

CloudFormation dukungan

Anda dapat menggunakan CloudFormation untuk meluncurkan cluster Neptunus dengan titik akhir publik diaktifkan dengan menentukan `PubliclyAccessible` parameter dalam template Anda. CloudFormation

Kompatibilitas dengan fitur Neptunus

Cluster dengan titik akhir publik yang diaktifkan mendukung semua fitur Neptunus yang didukung oleh cluster khusus VPC, termasuk:

- Meja kerja Neptunus
- Integrasi pencarian teks lengkap
- Pengaliran Neptune
- Titik akhir kustom
- Neptunus Tanpa Server
- Penjelajah Grafik

Harga

Titik akhir publik tersedia tanpa biaya tambahan di luar harga Neptunus standar. Namun, menghubungkan dari lingkungan lokal Anda ke Neptunus melalui IP publik mungkin menimbulkan peningkatan biaya transfer data.

Mengamankan akses ke cluster Amazon Neptunus

Ada beberapa cara bagi Anda untuk mengamankan klaster Amazon Neptune Anda.

Menggunakan kebijakan IAM untuk membatasi akses ke cluster DB Neptunus

Untuk mengontrol siapa yang dapat melakukan tindakan manajemen Neptunus pada cluster DB Neptunus dan instans DB, gunakan (IAM). AWS Identity and Access Management

Saat Anda menggunakan akun IAM untuk mengakses konsol Neptunus, Anda harus terlebih dahulu masuk ke Konsol Manajemen AWS akun menggunakan IAM Anda sebelum membuka konsol Neptunus di rumah. https://console.aws.amazon.com/neptune/

Saat Anda terhubung AWS menggunakan kredensil IAM, akun IAM Anda harus memiliki kebijakan IAM yang memberikan izin yang diperlukan untuk melakukan operasi manajemen Neptunus. Untuk informasi selengkapnya, lihat Menggunakan berbagai jenis kebijakan IAM untuk mengontrol akses ke Neptunus.

Menggunakan grup keamanan VPC untuk membatasi akses ke cluster DB Neptunus

Klaster DB Neptune harus dibuat di Amazon Virtual Private Cloud (Amazon VPC). Untuk mengontrol perangkat dan instans EC2 mana yang dapat membuka koneksi ke titik akhir dan port instans DB untuk klaster DB Neptune di VPC, Anda menggunakan grup keamanan VPC. Untuk informasi lebih lanjut tentang VPCs, lihatBuat grup keamanan menggunakan konsol VPC.

> **Note**
>
> Untuk terhubung ke cluster Neptunus Anda, Anda harus mengekspos port Database cluster (default 8182) untuk aturan masuk dan keluar untuk memungkinkan konektivitas yang tepat.

Menggunakan otentikasi IAM untuk membatasi akses ke cluster DB Neptunus

Jika Anda mengaktifkan otentikasi AWS Identity and Access Management (IAM) di cluster DB Neptunus, siapa pun yang mengakses cluster DB harus diautentikasi terlebih dahulu. Lihat Mengautentikasi database Amazon Neptunus Anda dengan AWS Identity and Access Management untuk informasi tentang pengaturan autentikasi IAM.

Untuk informasi tentang menggunakan kredensyal sementara untuk mengautentikasi, termasuk contoh untuk AWS CLI,, dan Amazon EC2 AWS Lambda, lihat. the section called “Kredensial sementara”

Tautan berikut memberikan informasi tambahan tentang menghubungkan ke Neptunus menggunakan otentikasi IAM dengan bahasa kueri individual:

Menggunakan Gremlin dengan otentikasi IAM

- the section called “Menghubungkan dari Gremlin Console”
- the section called “Menghubungkan menggunakan Gremlin Java”
- the section called “Menghubungkan dengan Python”

> **Note**
>
> Contoh ini berlaku untuk Gremlin dan SPARQL.

Menggunakan OpenCypher dengan otentikasi IAM

- the section called “Menghubungkan dari Gremlin Console”
- the section called “Menghubungkan menggunakan Gremlin Java”
- the section called “Menghubungkan dengan Python”

> **Note**
>
> Contoh ini berlaku untuk Gremlin dan SPARQL.

Menggunakan SPARQL dengan otentikasi IAM

- the section called “Terhubung dengan SPARQL Java”
- the section called “Menghubungkan dengan Python”

> **Note**
>
> Contoh ini berlaku untuk Gremlin dan SPARQL.

Mengakses data grafik di Amazon Neptunus

Anda dapat berinteraksi dengan cluster Amazon Neptunus DB setelah membuat koneksi. Ini melibatkan pemuatan data, mengeksekusi kueri, dan melakukan operasi lainnya. Sebagian besar pengguna memanfaatkan alat `awscurl` baris perintah `curl` atau untuk berkomunikasi dengan cluster DB Neptunus secara efektif. Alat-alat ini memungkinkan Anda mengirim permintaan, memuat data, dan mengambil hasil dari database grafik, memfasilitasi manajemen data yang mulus dan kemampuan kueri.

Menyiapkan **`curl`** untuk berkomunikasi dengan titik akhir Neptunus Anda

Seperti diilustrasikan dalam banyak contoh dalam dokumentasi ini, alat baris perintah curl adalah pilihan berguna untuk berkomunikasi dengan titik akhir Neptune Anda. Untuk informasi tentang alat tersebut, lihat halaman curl man, dan buku Semuanya curl.

Untuk menyambung menggunakan HTTPS (seperti yang kami sarankan dan seperti yang dibutuhkan Neptune di sebagian besar Wilayah), `curl` membutuhkan akses ke sertifikat yang sesuai. Untuk mempelajari cara mendapatkan sertifikat ini dan cara memformat mereka dengan benar ke penyimpanan sertifikat otoritas sertifikat (CA) yang dapat digunakan `curl`, lihat Verifikasi Sertifikat SSL dalam dokumentasi `curl`.

Anda kemudian dapat menentukan lokasi penyimpanan sertifikat CA ini menggunakan variabel lingkungan `CURL_CA_BUNDLE`. Pada Windows, `curl` secara otomatis mencarinya dalam sebuah file bernama `curl-ca-bundle.crt`. Ia pertama mencari dalam direktori yang sama dengan `curl.exe` dan kemudian di tempat lain di jalurnya. Untuk informasi lebih lanjut, lihat Verifikasi Sertifikat SSL.

Selama `curl` dapat menemukan sertifikat yang sesuai, ia menangani koneksi HTTPS seperti koneksi HTTP, tanpa parameter tambahan. Contoh dalam dokumentasi ini didasarkan pada skenario tersebut.

Menggunakan bahasa kueri untuk mengakses data grafik di cluster DB Neptunus Anda

Setelah Anda terhubung, Anda dapat menggunakan bahasa kueri Gremlin dan OpenCypher untuk membuat dan menanyakan grafik properti, atau bahasa kueri SPARQL untuk membuat dan menanyakan grafik yang berisi data RDF.

Bahasa kueri grafik yang didukung oleh Neptunus

- Gremlin adalah bahasa traversal grafik untuk grafik properti. Sebuah kueri di Gremlin adalah sebuah traversal yang terdiri dari langkah-langkah berlainan, yang masing-masing mengikuti edge ke simpul. Lihat dokumentasi Gremlin di Apache TinkerPop untuk informasi lebih lanjut.

  Implementasi Neptune dari Gremlin memiliki beberapa perbedaan dari implementasi lain, terutama ketika Anda menggunakan Gremlin-Groovy (kueri Gremlin dikirim sebagai teks serial). Untuk informasi selengkapnya, lihat Kepatuhan standar Gremlin di Amazon Neptune.
- OpenCypher adalah bahasa query deklaratif untuk grafik properti yang awalnya dikembangkan oleh Neo4j, kemudian open-source pada tahun 2015, dan berkontribusi pada proyek OpenCypher di bawah lisensi open-source Apache 2. Sintaksnya didokumentasikan dalam Referensi Bahasa Kueri Cypher, Versi 9.
- SPARQL adalah bahasa query deklaratif untuk data RDF, berdasarkan pencocokan pola grafik yang distandarisasi oleh World Wide Web Consortium (W3C) dan dijelaskan dalam SPARQL 1.1 Ikhtisar) dan spesifikasi SPARQL 1.1 Query Language.

> **Note**
>
> Anda dapat mengakses data grafik properti di Neptunus menggunakan Gremlin dan OpenCypher, tetapi tidak menggunakan SPARQL. Demikian pula, Anda hanya dapat mengakses data RDF menggunakan SPARQL, bukan Gremlin atau OpenCypher.

Menggunakan Gremlin untuk mengakses data grafik di Amazon Neptunus

Anda dapat menggunakan Konsol Gremlin untuk bereksperimen dengan TinkerPop grafik dan kueri di lingkungan REPL (loop). read-eval-print

Tutorial berikut memandu Anda melalui menggunakan konsol Gremlin untuk menambahkan vertex, edge, properti, dan banyak lagi ke grafik Neptune, menyoroti beberapa perbedaan dalam implementasi Gremlin khusus Neptune.

> **Note**
>
> Contoh ini mengasumsikan bahwa Anda telah menyelesaikan hal berikut ini:
>
> - Anda telah terhubung menggunakan SSH ke instans Amazon EC2.

- Anda telah membuat sebuah klaster Neptune seperti yang dijelaskan di Buat cluster Neptunus.
- Anda telah menginstal konsol Gremlin seperti yang diterangkan dalam Menginstal konsol Gremlin.

Menggunakan Konsol Gremlin

1. Ubah direktori ke dalam folder tempat file konsol Gremlin di-unzip.

```
cd apache-tinkerpop-gremlin-console-3.7.2
```

2. Masukkan perintah berikut untuk menjalankan Konsol Gremlin.

```
bin/gremlin.sh
```

   Anda akan melihat output berikut:

```
         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin>
```

   Anda sekarang berada di prompt `gremlin>`. Anda memasukkan langkah-langkah yang tersisa pada prompt ini.

3. Di prompt `gremlin>`, masukkan hal berikut untuk menyambung ke instans DB Neptune.

```
:remote connect tinkerpop.server conf/neptune-remote.yaml
```

4. Di prompt `gremlin>`, masukkan hal berikut ini untuk beralih ke mode jarak jauh. Ini mengirimkan semua kueri Gremlin ke koneksi remote.

```
:remote console
```

5. Tambahkan vertex dengan label dan properti.

```
g.addV('person').property('name', 'justin')
```

Vertex ditugaskan ID `string` yang berisi GUID. Semua simpul IDs adalah string di Neptunus.

6. Tambahkan simpul dengan id kustom.

```
g.addV('person').property(id, '1').property('name', 'martin')
```

Properti `id` tidak dikutip. Ini adalah kata kunci untuk ID vertex. ID vertex di sini adalah string dengan nomor 1 di dalamnya.

Nama properti normal harus berada di dalam tanda kutip.

7. Ubah properti atau tambahkan properti jika tidak ada.

```
g.V('1').property(single, 'name', 'marko')
```

Di sini Anda mengubah properti `name` untuk vertex dari langkah sebelumnya. Ini akan menghapus semua nilai yang ada dari properti `name`.

Jika Anda tidak menentukan `single`, ia justru akan menambahkan nilainya ke properti `name` jika belum dilakukannya.

8. Tambahkan properti, tetapi tambahkan properti jika properti sudah memiliki nilai.

```
g.V('1').property('age', 29)
```

Neptune menggunakan set secara kardinal sebagai tindakan default.

Perintah ini menambahkan properti `age` dengan nilai 29, tetapi tidak menggantikan nilai yang ada.

Jika properti `age` sudah memiliki nilai, perintah ini menambahkan 29 ke properti. Sebagai contoh, jika properti `age` adalah 27, nilai baru akan menjadi `[ 27, 29 ]`.

9. Tambahkan beberapa simpul.

```
g.addV('person').property(id, '2').property('name', 'vadas').property('age',
 27).iterate()
g.addV('software').property(id, '3').property('name', 'lop').property('lang',
 'java').iterate()
```

```
g.addV('person').property(id, '4').property('name', 'josh').property('age',
 32).iterate()
g.addV('software').property(id, '5').property('name', 'ripple').property('lang',
 'java').iterate()
g.addV('person').property(id, '6').property('name', 'peter').property('age', 35)
```

Anda dapat mengirim beberapa pernyataan pada saat yang sama ke Neptune.

Pernyataan dapat dipisahkan dengan baris baru (`'\n'`), spasi (`' '`), titik koma (`'; '`), atau kosong (misalnya: `g.addV('person').iterate()g.V()` berlaku).

> **Note**
>
> Konsol Gremlin mengirimkan perintah terpisah di setiap baris baru (`'\n'`), sehingga masing-masing mereka menjadi transaksi terpisah dalam kasus itu. Contoh ini memiliki semua perintah pada baris terpisah untuk dibaca. Hapus baris baru karakter (`'\n'`) untuk mengirimkannya sebagai perintah tunggal melalui Konsol Gremlin.

Semua pernyataan selain pernyataan terakhir harus diakhiri dengan langkah pengakhiran, seperti `.next()` atau `.iterate()`, atau mereka tidak akan berjalan. Konsol Gremlin tidak memerlukan langkah-langkah pengakhiran ini. Gunakan `.iterate` setiap kali Anda tidak memerlukan hasil yang akan diserialkan.

Semua pernyataan yang dikirim bersama-sama disertakan dalam satu transaksi dan berhasil atau gagal bersama-sama.

10. Tambahkan tepi.

```
g.V('1').addE('knows').to(__.V('2')).property('weight', 0.5).iterate()
g.addE('knows').from(__.V('1')).to(__.V('4')).property('weight', 1.0)
```

Berikut adalah dua cara berbeda untuk menambahkan edge.

11. Tambahkan sisa grafik Modern.

```
g.V('1').addE('created').to(__.V('3')).property('weight', 0.4).iterate()
g.V('4').addE('created').to(__.V('5')).property('weight', 1.0).iterate()
g.V('4').addE('knows').to(__.V('3')).property('weight', 0.4).iterate()
g.V('6').addE('created').to(__.V('3')).property('weight', 0.2)
```

12. Hapus simpul.

```
g.V().has('name', 'justin').drop()
```

Menghapus vertex dengan properti `name` sama dengan `justin`.

> **Important**
>
> Berhenti di sini, dan Anda memiliki grafik Apache TinkerPop Modern lengkap. Contoh di bagian Traversal TinkerPop dokumentasi menggunakan grafik Modern.

13. Jalankan traversal.

```
g.V().hasLabel('person')
```

Mengembalikan semua vertex `person`.

14. Jalankan Traversal dengan nilai (valueMap ()).

```
g.V().has('name', 'marko').out('knows').valueMap()
```

Mengembalikan kunci, pasangan nilai untuk semua vertex yang “diketahui” `marko`.

15. Tentukan beberapa label.

```
g.addV("Label1::Label2::Label3")
```

Neptune mendukung beberapa label untuk sebuah vertex. Ketika Anda membuat label, Anda dapat menentukan beberapa label dengan memisahkannya dengan `::`.

Contoh ini menambahkan sebuah vertex dengan tiga label yang berbeda.

Langkah `hasLabel` cocok dengan vertex ini dengan salah satu dari tiga label tersebut: `hasLabel("Label1")`, `hasLabel("Label2")`, dan `hasLabel("Label3")`.

Pembatas `::` dicadangkan untuk penggunaan ini saja.

Anda tidak dapat menentukan beberapa label di langkah `hasLabel`. Misalnya, `hasLabel("Label1::Label2")` tidak cocok dengan apa pun.

16. Tentukan Waktu/tanggal.

```
g.V().property(single, 'lastUpdate', datetime('2018-01-01T00:00:00'))
```

Neptune tidak mendukung Java Date. Gunakan `datetime()` fungsi sebagai gantinya. `datetime()`menerima string ISO8061 `datetime` -compliant.

Mendukung format berikut: `YYYY-MM-DD`, `YYYY-MM-DDTHH:mm`, `YYYY-MM-DDTHH:mm:SS`, dan `YYYY-MM-DDTHH:mm:SSZ`.

17. Hapus simpul, properti, atau tepi.

```
g.V().hasLabel('person').properties('age').drop().iterate()
g.V('1').drop().iterate()
g.V().outE().hasLabel('created').drop()
```

Berikut adalah beberapa contoh drop.

> **Note**
>
> Langkah `.next()` tidak bekerja dengan `.drop()`. Gunakan `.iterate()` sebagai gantinya.

18. Setelah selesai, masukkan yang berikut ini untuk keluar dari Gremlin Console.

```
:exit
```

> **Note**
>
> Gunakan titik koma (`;`) atau karakter baris baru (`\n`) untuk memisahkan setiap pernyataan. Setiap traversal sebelum traversal akhir harus diakhiri dengan `iterate()` yang akan dieksekusi. Hanya data dari traversal akhir yang dikembalikan.

Menggunakan OpenCypher untuk mengakses data grafik di Amazon Neptunus

Untuk mulai menggunakan OpenCypher, lihatOpenCypher, atau gunakan notebook OpenCypher di repositori graf-notebook Neptunus. GitHub

Menggunakan SPARQL untuk mengakses data grafik di Amazon Neptunus

SPARQL adalah bahasa kueri untuk Resource Description Framework (RDF), yang merupakan format data grafik yang dirancang untuk web. Amazon Neptune kompatibel dengan SPARQL 1.1. Ini berarti bahwa Anda dapat terhubung ke instans DB Neptune dan mengajukan kueri grafik menggunakan bahasa kueri yang dijelaskan dalam spesifikasi Bahasa Kueri SPARQL 1.1.

Sebuah kueri di SPARQL terdiri dari klausa `SELECT` untuk menentukan variabel yang akan dikembalikan dan klausa `WHERE` untuk menentukan data yang mana yang akan dicocokkan dalam grafik. Jika Anda tidak terbiasa dengan kueri SPARQL, lihat Menulis Kueri Sederhana dalam Bahasa Kueri SPARQL 1.1.

Titik akhir HTTP untuk kueri SPARQL ke instans DB Neptune adalah: `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.

Untuk terhubung ke SPARQL

1. Anda bisa mendapatkan titik akhir SPARQL untuk cluster Neptunus Anda dari SparqlEndpointitem di bagian Output tumpukan. CloudFormation
2. Masukkan hal berikut untuk mengirimkan **`UPDATE`** SPARQL menggunakan HTTP `POST` dan perintah curl.

   ```
   curl -X POST --data-binary 'update=INSERT DATA { <https://test.com/s> <https://
   test.com/p> <https://test.com/o> . }' https://your-neptune-endpoint:port/sparql
   ```

   Contoh sebelumnya menyisipkan tripel berikut ke dalam grafik default SPARQL: `<https://test.com/s> <https://test.com/p> <https://test.com/o>`
3. Masukkan hal berikut untuk mengirimkan **`QUERY`** SPARQL menggunakan HTTP `POST` dan perintah curl.

   ```
   curl -X POST --data-binary 'query=select ?s ?p ?o where {?s ?p ?o} limit 10'
    https://your-neptune-endpoint:port/sparql
   ```

   Contoh sebelumnya mengembalikan hingga 10 dari triple (subject-predicate-object) dalam grafik dengan menggunakan `?s ?p ?o` query dengan batas 10. Untuk mengajukan kueri untuk sesuatu yang lain, gantikan dengan kueri SPARQL lain.

> **Note**
>
> Jenis MIME default respon adalah `application/sparql-results+json` untuk kueri `SELECT` dan `ASK`.
> Jenis MIME default respon adalah `application/n-quads` untuk kueri `CONSTRUCT` dan `DESCRIBE`.
> Untuk daftar semua jenis MIME yang tersedia, lihat API HTTP SPARQL.

Memuat Data ke dalam cluster Amazon Neptunus

Amazon Neptune menyediakan proses untuk memuat data dari file eksternal langsung ke instans DB Neptune. Anda dapat menggunakan proses ini alih-alih mengeksekusi sejumlah besar pernyataan `INSERT`, langkah `addV` dan `addE`, atau panggilan API lainnya.

Berikut ini adalah link ke informasi pemuatan tambahan.

- Metode untuk memuat data - Memuat data
- Format data yang didukung oleh pemuat massal - the section called “Format Data”
- Contoh pemuatan — the section called “Contoh Pemuatan”

Memantau cluster Amazon Neptunus

Amazon Neptune mendukung metode pemantauan berikut.

- Amazon CloudWatch - Amazon Neptunus secara otomatis mengirimkan metrik CloudWatch ke dan juga mendukung Alarm. CloudWatch Untuk informasi selengkapnya, lihat the section called “Menggunakan CloudWatch”.
- AWS CloudTrail- Amazon Neptunus mendukung pencatatan API menggunakan. CloudTrail Untuk informasi selengkapnya, lihat the section called “Mencatat Panggilan API Neptunus dengan AWS CloudTrail”.
- Penandaan— Gunakan tanda untuk menambahkan metadata ke sumber daya Neptune Anda dan lacak penggunaan berdasarkan tanda. Untuk informasi selengkapnya, lihat the section called “Menandai sumber daya Neptunus”.

- File log audit— Lihat, unduh, atau tonton file log basis data menggunakan konsol Neptune. Untuk informasi selengkapnya, lihat the section called “Log Audit dengan Neptune”.
- Status instans — Periksa kesehatan mesin database grafik instans Neptune, cari tahu versi mesin yang diinstal, dan dapatkan informasi status mesin lainnya menggunakan API status instans.

Memecahkan masalah klaster Amazon Neptunus

Tautan berikut mungkin berguna untuk menyelesaikan masalah dengan Amazon Neptune.

- Praktik terbaik— Untuk solusi masalah umum dan saran kinerja, lihat Praktik terbaik.
- Kesalahan layanan — Untuk daftar kesalahan untuk koneksi database manajemen APIs dan grafik, lihatKesalahan Neptune.
- Batas layanan— Untuk informasi tentang batas Neptune, lihat Batas Neptune.
- Rilis mesin— Untuk informasi tentang rilis mesin grafik, termasuk catatan rilis, lihat Rilis mesin.
- Forum Support— Untuk bergabung dalam diskusi tentang Neptune, lihat Forum Amazon Neptune.
- Harga— Untuk informasi tentang biaya penggunaan Amazon Neptune, lihat Harga Amazon Neptune.
- AWS Support — Untuk bantuan dan bimbingan dari para ahli, lihat AWS Dukungan.

Amazon Neptunus Tanpa Server

Amazon Neptune Serverless adalah konfigurasi penskalaan otomatis sesuai permintaan yang dirancang untuk menskalakan cluster DB Anda sesuai kebutuhan untuk memenuhi peningkatan permintaan pemrosesan yang sangat besar, dan kemudian menurunkan skala lagi ketika permintaan menurun. Ini membantu mengotomatiskan proses pemantauan beban kerja dan menyesuaikan kapasitas untuk database Neptunus Anda. Karena kapasitas disesuaikan secara otomatis berdasarkan permintaan aplikasi, Anda hanya dikenakan biaya untuk sumber daya yang sebenarnya dibutuhkan aplikasi Anda.

Kasus penggunaan untuk Neptunus Tanpa Server

Neptune Serverless mendukung banyak jenis beban kerja. Ini cocok untuk beban kerja yang menuntut dan sangat bervariasi dan dapat sangat membantu jika penggunaan database Anda biasanya berat untuk waktu yang singkat, diikuti oleh periode aktivitas ringan yang lama atau tidak ada aktivitas sama sekali. Neptunus Tanpa Server sangat berguna untuk kasus penggunaan berikut:

- Beban kerja variabel — Beban kerja yang memiliki peningkatan aktivitas CPU yang tiba-tiba dan tidak dapat diprediksi. Dengan Neptunus Tanpa Server, database grafik Anda secara otomatis menskalakan kapasitas untuk memenuhi kebutuhan beban kerja dan menurunkan skala saat lonjakan aktivitas selesai. Anda tidak lagi harus menyediakan kapasitas puncak atau rata-rata. Anda dapat menentukan batas kapasitas atas untuk menangani beban kerja puncak, dan kapasitas itu tidak digunakan kecuali diperlukan.

  Perincian penskalaan yang disediakan oleh Neptune Serverless membantu Anda mencocokkan kapasitas dengan kebutuhan beban kerja Anda. Neptunus Tanpa Server dapat menambah atau menghapus kapasitas dalam peningkatan berbutir halus berdasarkan apa yang dibutuhkan. Ini dapat menambahkan sesedikit setengah Unit Kapasitas Neptunus (NCU) ketika hanya sedikit lebih banyak kapasitas yang diperlukan.
- Aplikasi multi-tenant — Dengan memanfaatkan Neptunus Tanpa Server, Anda dapat membuat cluster DB terpisah untuk setiap aplikasi yang perlu Anda jalankan tanpa harus mengelola cluster penyewa tersebut secara individual. Setiap cluster penyewa mungkin memiliki periode sibuk dan idle yang berbeda tergantung pada beberapa faktor, tetapi Neptunus Tanpa Server dapat menskalakannya secara efisien tanpa campur tangan Anda.
- Aplikasi baru — Ketika Anda menerapkan aplikasi baru, Anda sering tidak yakin berapa banyak kapasitas database yang dibutuhkan. Menggunakan Neptunus Tanpa Server, Anda dapat

mengatur cluster DB yang dapat menskalakan secara otomatis untuk memenuhi persyaratan kapasitas aplikasi baru saat mereka berkembang.

- Perencanaan kapasitas - Misalkan Anda biasanya menyesuaikan kapasitas database Anda, atau memverifikasi kapasitas database optimal untuk beban kerja Anda, dengan memodifikasi kelas instans DB dari semua instans DB dalam sebuah cluster. Dengan Neptunus Tanpa Server, Anda dapat menghindari overhead administratif ini. Sebagai gantinya, Anda dapat memodifikasi instans DB yang ada dari yang disediakan ke tanpa server atau dari tanpa server ke yang disediakan tanpa harus membuat cluster atau instance DB baru.
- Pengembangan dan pengujian - Neptune Serverless juga sempurna untuk lingkungan pengembangan dan pengujian. Dengan Neptune Serverless, Anda dapat membuat instans DB dengan kapasitas maksimum yang cukup tinggi untuk menguji aplikasi Anda yang paling menuntut, dan kapasitas minimum yang rendah untuk semua waktu lain ketika sistem mungkin menganggur di antara pengujian.

Neptunus Tanpa Server hanya menskalakan kapasitas komputasi. Volume penyimpanan Anda tetap sama, dan tidak terpengaruh oleh penskalaan tanpa server.

> **Note**
>
> Anda juga dapat menggunakan auto-scaling Neptunus dengan Neptune Serverless untuk menangani berbagai jenis variasi beban kerja.

Kendala Amazon Neptunus Tanpa Server

- Neptunus Tanpa Server hanya tersedia di wilayah berikut:
  - US East (N. Virginia): `us-east-1`
  - AS Timur (Ohio): `us-east-2`
  - US West (N. California): `us-west-1`
  - US West (Oregon): `us-west-2`
  - Canada (Central): `ca-central-1`
  - Eropa (Stockholm): `eu-north-1`
  - Eropa (Spanyol): `eu-south-2`
  - Eropa (Irlandia): `eu-west-1`

- Eropa (London): `eu-west-2`
- Eropa (Paris): `eu-west-3`
- Eropa (Frankfurt): `eu-central-1`
- Asia Pacific (Tokyo): `ap-northeast-1`
- Asia Pasifik (Seoul): `ap-northeast-2`
- Asia Pacific (Singapore): `ap-southeast-1`
- Asia Pacific (Sydney): `ap-southeast-2`
- Asia Pasifik (Jakarta): `ap-southeast-3`
- Asia Pasifik (Hong Kong): `ap-east-1`
- Asia Pasifik (Mumbai): `ap-south-1`
- South America (São Paulo): `sa-east-1`

- Tidak tersedia dalam versi mesin awal — Neptune Serverless hanya tersedia di rilis mesin 1.2.0.1 atau yang lebih baru.
- Tidak kompatibel dengan cache pencarian Neptunus — Cache pencarian tidak berfungsi dengan instans DB tanpa server.
- Memori maksimum dalam instance tanpa server adalah 256 GB - Pengaturan `MaxCapacity` ke 128 NCUs (pengaturan tertinggi yang didukung) memungkinkan instans Neptunus Tanpa Server untuk menskalakan memori 256 GB, yang setara dengan jenis instans yang disediakan. `R6g.8XL`

Penskalaan kapasitas dalam cluster DB Neptunus Tanpa Server

Menyiapkan cluster DB Tanpa Server Neptunus mirip dengan menyiapkan cluster yang disediakan normal, dengan konfigurasi tambahan untuk unit minimum dan maksimum untuk penskalaan, dan dengan jenis instans disetel ke. `db.serverless` Konfigurasi penskalaan didefinisikan dalam Unit Kapasitas Neptunus NCUs (), yang masing-masing terdiri dari 2 GiB (gibibyte) memori (RAM) bersama dengan kapasitas prosesor virtual terkait (vCPU) dan jaringan. Hal ini diatur sebagai bagian dari `ServerlessV2ScalingConfiguration` objek, diwakili dalam JSON seperti ini:

```
"ServerlessV2ScalingConfiguration": {
  "MinCapacity": (minimum NCUs, a floating-point number such as   1.0),
  "MaxCapacity": (maximum NCUs, a floating-point number such as 128.0)
}
```

Setiap saat, setiap penulis atau pembaca Neptunus memiliki kapasitas yang diukur dengan angka floating-point yang mewakili jumlah saat ini digunakan oleh instance itu. NCUs Anda dapat menggunakan CloudWatch ServerlessDatabaseCapacitymetrik pada tingkat instans untuk mengetahui berapa banyak instans DB NCUs yang digunakan saat ini, dan NCUUtilizationmetrik untuk mengetahui berapa persentase kapasitas maksimumnya yang digunakan instans. Kedua metrik ini juga tersedia pada tingkat cluster DB untuk menunjukkan pemanfaatan sumber daya rata-rata untuk cluster DB secara keseluruhan.

Saat membuat cluster DB Neptunus Tanpa Server, Anda menetapkan jumlah minimum dan maksimum unit kapasitas Neptunus () untuk semua instance tanpa server. NCUs

Nilai NCU minimum yang Anda tentukan menetapkan ukuran terkecil yang dapat menyusut instans tanpa server di kluster DB Anda, dan juga, nilai NCU maksimum menetapkan ukuran terbesar tempat instance tanpa server dapat tumbuh. Nilai NCU maksimum tertinggi yang dapat Anda atur adalah 128,0 NCUs, dan minimum terendah adalah 1,0. NCUs

Neptunus terus melacak beban pada setiap instance Neptunus Tanpa Server dengan memantau pemanfaatan sumber daya seperti CPU, memori, dan jaringan. Beban dihasilkan oleh operasi database aplikasi Anda, dengan pemrosesan latar belakang untuk server, dan oleh tugas administratif lainnya.

Ketika beban pada instance tanpa server mencapai batas kapasitas saat ini, atau saat Neptunus mendeteksi masalah kinerja lainnya, instans akan meningkat secara otomatis. Ketika beban pada instans menurun, kapasitas akan turun ke unit kapasitas minimum yang dikonfigurasi, dengan kapasitas CPU dilepaskan sebelum memori. Arsitektur ini memungkinkan pelepasan sumber daya dengan cara step-down yang terkontrol dan menangani fluktuasi permintaan secara efektif.

Anda dapat membuat skala instance pembaca bersama dengan instance penulis atau skala secara independen dengan menetapkan tingkat promosinya. Contoh pembaca dalam skala promosi 0 dan 1 pada saat yang sama dengan penulis, yang membuat mereka berukuran pada kapasitas yang tepat untuk mengambil alih beban kerja dari penulis dengan cepat jika terjadi kegagalan. Pembaca di tingkat promosi 2 hingga 15 skala independen dari contoh penulis, dan satu sama lain.

Jika Anda telah membuat cluster DB Neptunus sebagai klaster multi-AZ untuk memastikan ketersediaan tinggi, Neptune Serverless menskalakan instans secara naik dan turun dengan pemuatan basis data Anda. AZs Anda dapat mengatur tingkat promosi instance pembaca di AZ sekunder ke 0 atau 1 sehingga skala naik dan turun bersama dengan kapasitas instance penulis di AZ utama sehingga siap untuk mengambil alih beban kerja saat ini kapan saja.

> **Note**
>
> Penyimpanan untuk cluster DB Neptunus terdiri dari enam salinan dari semua data Anda, tersebar di AZs tiga, terlepas dari apakah Anda membuat cluster sebagai cluster multi-AZ atau tidak. Replikasi penyimpanan ditangani oleh subsistem penyimpanan dan tidak terpengaruh oleh Neptunus Tanpa Server.

Memilih nilai kapasitas minimum untuk cluster DB Neptunus Tanpa Server

Nilai terkecil yang dapat Anda atur untuk kapasitas minimum adalah `1.0` NCUs.

Pastikan untuk tidak menetapkan nilai minimum lebih rendah dari yang dibutuhkan aplikasi Anda untuk beroperasi secara efisien. Mengaturnya terlalu rendah dapat menghasilkan tingkat batas waktu yang lebih tinggi dalam beban kerja intensif memori tertentu.

Menetapkan nilai minimum serendah mungkin dapat menghemat uang, karena cluster Anda akan menggunakan sumber daya minimal saat permintaan rendah. Namun, jika beban kerja Anda cenderung berfluktuasi secara dramatis, dari sangat rendah hingga sangat tinggi, Anda mungkin ingin menetapkan minimum yang lebih tinggi, karena minimum yang lebih tinggi memungkinkan instans Neptunus Tanpa Server Anda meningkat lebih cepat.

Alasan untuk ini adalah bahwa Neptunus memilih peningkatan skala berdasarkan kapasitas saat ini. Jika kapasitas saat ini rendah, Neptunus awalnya akan meningkat secara perlahan. Jika minimum lebih tinggi, Neptunus dimulai dengan peningkatan skala yang lebih besar, dan karenanya dapat meningkatkan skala lebih cepat untuk memenuhi peningkatan beban kerja yang tiba-tiba.

Memilih nilai kapasitas maksimum untuk cluster DB Neptunus Tanpa Server

Nilai terbesar yang dapat Anda atur untuk kapasitas maksimum adalah `128.0` NCUs, dan nilai terkecil yang dapat Anda atur untuk kapasitas maksimum adalah `2.5` NCUs. Apapun nilai kapasitas maksimum yang Anda tetapkan harus setidaknya sebesar nilai kapasitas minimum yang Anda tetapkan.

Sebagai aturan umum, tetapkan nilai maksimum yang cukup tinggi untuk menangani beban puncak yang mungkin dihadapi aplikasi Anda. Mengaturnya terlalu rendah dapat menghasilkan tingkat batas waktu yang lebih tinggi dalam beban kerja intensif memori tertentu.

Menetapkan nilai maksimum setinggi mungkin memiliki keuntungan bahwa aplikasi Anda kemungkinan besar dapat menangani beban kerja yang paling tidak terduga sekalipun. Kerugiannya

adalah Anda kehilangan beberapa kemampuan untuk memprediksi dan mengendalikan biaya sumber daya. Lonjakan permintaan yang tidak terduga dapat berakhir dengan biaya lebih dari yang diperkirakan anggaran Anda.

Manfaat dari nilai maksimum yang ditargetkan dengan hati-hati adalah memungkinkan Anda memenuhi permintaan puncak sambil juga membatasi biaya komputasi Neptunus.

> **Note**
>
> Mengubah rentang kapasitas cluster DB Neptunus Tanpa Server menyebabkan perubahan pada nilai default dari beberapa parameter konfigurasi. Neptunus dapat segera menerapkan beberapa default baru tersebut, tetapi beberapa perubahan parameter dinamis hanya berlaku setelah reboot. `pending-reboot`Status menunjukkan bahwa Anda memerlukan reboot untuk menerapkan beberapa perubahan parameter.

Gunakan konfigurasi yang ada untuk memperkirakan persyaratan tanpa server

Jika Anda biasanya memodifikasi kelas instans DB dari instans DB yang disediakan untuk memenuhi beban kerja yang sangat tinggi atau rendah, Anda dapat menggunakan pengalaman itu untuk membuat perkiraan kasar rentang kapasitas Neptunus Tanpa Server yang setara.

Perkirakan pengaturan kapasitas minimum terbaik

Anda dapat menerapkan apa yang Anda ketahui tentang cluster DB Neptunus yang ada untuk memperkirakan pengaturan kapasitas minimum tanpa server yang akan bekerja paling baik.

Misalnya, jika beban kerja Anda yang disediakan memiliki persyaratan memori yang terlalu tinggi untuk kelas instans DB kecil seperti `T3` atau`T4g`, pilih pengaturan NCU minimum yang menyediakan memori yang sebanding dengan kelas instans `R5` atau `R6g` DB.

Atau, misalkan Anda menggunakan kelas instans `db.r6g.xlarge` DB ketika cluster Anda memiliki beban kerja yang rendah. Kelas instans DB itu memiliki memori 32 GiB, sehingga Anda dapat menentukan pengaturan NCU minimum 16 untuk membuat instance tanpa server yang dapat menurunkan skala hingga kira-kira kapasitas yang sama (setiap NCU sesuai dengan sekitar 2 GiB memori). Jika `db.r6g.xlarge` instance Anda terkadang kurang dimanfaatkan, Anda mungkin dapat menentukan nilai yang lebih rendah.

Jika aplikasi Anda bekerja paling efisien ketika instans DB Anda dapat menyimpan sejumlah data tertentu dalam memori atau cache buffer, pertimbangkan untuk menentukan pengaturan NCU minimum yang cukup besar untuk menyediakan memori yang cukup untuk itu. Jika tidak, data dapat dikeluarkan dari cache buffer saat instance tanpa server menurunkan skala, dan harus dibaca kembali ke cache buffer dari waktu ke waktu ketika instance diskalakan cadangan. Jika jumlah I/O untuk membawa data kembali ke cache buffer sangat besar, memilih nilai NCU minimum yang lebih tinggi bisa bermanfaat.

Jika Anda menemukan bahwa instans tanpa server Anda berjalan sebagian besar waktu pada kapasitas tertentu, itu berfungsi dengan baik untuk mengatur kapasitas minimum hanya sedikit lebih rendah dari itu. Neptunus Tanpa Server dapat secara efisien memperkirakan berapa banyak dan seberapa cepat untuk ditingkatkan ketika kapasitas saat ini tidak secara drastis lebih rendah dari kapasitas yang dibutuhkan.

Dalam konfigurasi campuran, dengan penulis yang disediakan dan pembaca Neptunus Tanpa Server, pembaca tidak berskala bersama penulis. Karena mereka menskalakan secara independen, pengaturan kapasitas minimum yang rendah untuk mereka dapat mengakibatkan kelambatan replikasi yang berlebihan. Mereka mungkin tidak memiliki kapasitas yang cukup untuk mengikuti perubahan yang dibuat penulis ketika ada beban kerja yang sangat intensif menulis. Dalam hal ini, tetapkan kapasitas minimum yang sebanding dengan kapasitas penulis. Khususnya, jika Anda mengamati kelambatan replika pada pembaca yang berada di tingkatan promosi 2—15, tingkatkan pengaturan kapasitas minimum untuk klaster Anda.

Perkirakan pengaturan kapasitas maksimum terbaik

Anda juga dapat menerapkan apa yang Anda ketahui tentang cluster DB Neptunus yang ada untuk memperkirakan pengaturan kapasitas maksimum tanpa server yang akan bekerja paling baik.

Misalnya, Anda menggunakan kelas instans `db.r6g.4xlarge` DB ketika cluster Anda memiliki beban kerja yang tinggi. Kelas instans DB itu memiliki memori 128 GiB, sehingga Anda dapat menentukan pengaturan NCU maksimum 64 untuk mengatur instance Tanpa Server Neptunus yang setara (setiap NCU sesuai dengan sekitar 2 GiB memori). Anda dapat menentukan nilai yang lebih tinggi agar instans DB ditingkatkan lebih lanjut jika `db.r6g.4xlarge` instans Anda tidak selalu dapat menangani beban kerja.

Jika lonjakan tak terduga dalam beban kerja Anda jarang terjadi, mungkin masuk akal untuk mengatur kapasitas maksimum Anda cukup tinggi untuk mempertahankan kinerja aplikasi bahkan selama lonjakan tersebut. Di sisi lain, Anda mungkin ingin menetapkan kapasitas maksimum yang lebih rendah yang dapat mengurangi throughput selama lonjakan yang tidak biasa tetapi itu

memungkinkan Neptunus untuk menangani beban kerja yang Anda harapkan tanpa masalah, dan itu membatasi biaya.

Konfigurasi tambahan untuk cluster dan instance DB Neptunus Tanpa Server

Selain mengatur kapasitas minimum dan maksimum untuk cluster DB Neptunus Tanpa Server Anda, ada beberapa pilihan konfigurasi lain yang perlu dipertimbangkan.

Menggabungkan instans tanpa server dan yang disediakan dalam cluster DB

Cluster DB tidak harus tanpa server saja— Anda dapat membuat kombinasi instance tanpa server dan yang disediakan (konfigurasi campuran).

Misalnya, anggaplah Anda membutuhkan lebih banyak kapasitas tulis daripada yang tersedia dalam instance tanpa server. Dalam hal ini, Anda dapat mengatur cluster dengan penulis yang disediakan sangat besar dan masih menggunakan instance tanpa server untuk pembaca.

Atau, anggaplah beban kerja tulis pada klaster Anda bervariasi tetapi beban kerja baca stabil. Dalam hal ini, Anda dapat mengatur cluster Anda dengan penulis tanpa server dan satu atau lebih pembaca yang disediakan.

Lihat Menggunakan Amazon Neptune Tanpa Server untuk informasi tentang cara membuat cluster DB konfigurasi campuran.

Menyetel tingkatan promosi untuk instans Neptunus Tanpa Server

Untuk klaster yang berisi beberapa instance tanpa server, atau campuran instance yang disediakan dan tanpa server, perhatikan setelan tingkat promosi untuk setiap instance tanpa server. Setelan ini mengontrol lebih banyak perilaku untuk instance tanpa server daripada instans DB yang disediakan.

Dalam Konsol Manajemen AWS, Anda menentukan pengaturan ini menggunakan prioritas Failover di bawah Konfigurasi tambahan pada Buat database, Modifikasi instance, dan Tambahkan halaman pembaca. Anda melihat properti ini untuk instance yang ada di kolom tingkat Prioritas opsional pada halaman Database. Anda juga dapat melihat properti ini di halaman detail untuk cluster atau instance DB.

Untuk contoh yang disediakan, pilihan tingkat 0-15 hanya menentukan urutan di mana Neptunus memilih instance pembaca mana yang akan dipromosikan kepada penulis selama operasi failover. Untuk instance pembaca Neptunus Tanpa Server, nomor tingkatan juga menentukan apakah instans menskalakan agar sesuai dengan kapasitas instance penulis atau menskalakannya secara independen hanya berdasarkan beban kerjanya sendiri.

Contoh pembaca Neptunus Tanpa Server di tingkat 0 atau 1 disimpan pada kapasitas minimum setidaknya setinggi instance penulis sehingga mereka siap untuk mengambil alih dari penulis jika terjadi failover. Jika penulis adalah instance yang disediakan, Neptunus memperkirakan kapasitas tanpa server yang setara dan menggunakan estimasi itu sebagai kapasitas minimum untuk instance pembaca tanpa server.

Contoh pembaca Neptunus Tanpa Server di tingkatan 2-15 tidak memiliki batasan yang sama pada kapasitas minimumnya, dan skala secara independen dari penulis. Ketika mereka menganggur, mereka menurunkan ke nilai NCU minimum yang ditentukan dalam rentang kapasitas cluster. Namun, ini dapat menyebabkan masalah jika beban kerja baca melonjak dengan cepat.

Menjaga kapasitas pembaca selaras dengan kapasitas penulis

Satu hal penting yang perlu diingat adalah bahwa Anda ingin memastikan instance pembaca Anda dapat mengikuti contoh penulis Anda, untuk mencegah kelambatan replikasi yang berlebihan. Ini terutama menjadi perhatian dalam dua situasi, di mana instance pembaca tanpa server tidak secara otomatis disinkronkan dengan instance penulis:

- Ketika penulis Anda disediakan, dan pembaca Anda tanpa server.
- Ketika penulis Anda tanpa server, dan pembaca tanpa server Anda berada di tingkat promosi 2-15.

Dalam kedua kasus tersebut, tetapkan kapasitas tanpa server minimum agar sesuai dengan kapasitas penulis yang diharapkan, untuk memastikan bahwa operasi pembaca tidak habis waktu dan berpotensi menyebabkan restart. Dalam kasus instance penulis yang disediakan, tetapkan kapasitas minimum agar sesuai dengan instance yang disediakan. Dalam kasus penulis tanpa server, pengaturan optimal mungkin lebih sulit diprediksi.

Karena rentang kapasitas instans diatur pada tingkat cluster, semua instans tanpa server dikendalikan oleh pengaturan kapasitas minimum dan maksimum yang sama. Contoh pembaca dalam skala tingkatan 0 dan 1 selaras dengan instance penulis, tetapi contoh dalam tingkatan promosi skala 2-15 secara independen satu sama lain dan dari contoh penulis, tergantung pada beban kerja mereka. Jika Anda menyetel kapasitas minimum terlalu rendah, instans idle di tingkatan

2 hingga 15 dapat menurunkan skala terlalu rendah untuk menskalakan cadangan cukup cepat untuk menangani ledakan tiba-tiba dalam aktivitas penulis.

Hindari pengaturan nilai batas waktu terlalu tinggi

Dimungkinkan untuk mengeluarkan biaya tak terduga jika Anda menetapkan nilai batas waktu kueri terlalu tinggi pada instance tanpa server.

Tanpa pengaturan batas waktu yang wajar, Anda mungkin secara tidak sengaja mengeluarkan kueri yang membutuhkan jenis instans yang kuat dan mahal dan yang terus berjalan untuk waktu yang sangat lama, menimbulkan biaya yang tidak pernah Anda antisipasi. Anda dapat menghindari situasi itu dengan menggunakan nilai batas waktu kueri yang mengakomodasi sebagian besar kueri Anda dan hanya menyebabkan yang berjalan lama secara tak terduga habis.

Hal ini berlaku baik untuk nilai batas waktu kueri umum yang disetel menggunakan parameter dan untuk nilai batas waktu per kueri yang ditetapkan menggunakan petunjuk kueri.

Mengoptimalkan konfigurasi Neptunus Tanpa Server

Jika cluster DB Neptunus Tanpa Server Anda tidak disetel ke beban kerja yang dijalankannya, Anda mungkin memperhatikan bahwa itu tidak berjalan secara optimal. Anda dapat menyesuaikan pengaturan kapasitas and/or maksimum minimum sehingga dapat skala tanpa mengalami masalah memori.

- Tingkatkan pengaturan kapasitas minimum untuk cluster. Ini dapat memperbaiki situasi di mana instans idle menskalakan kembali ke kapasitas yang memiliki memori lebih sedikit daripada yang dibutuhkan aplikasi dan fitur yang diaktifkan.
- Tingkatkan pengaturan kapasitas maksimum untuk cluster. Ini dapat memperbaiki situasi di mana database yang sibuk tidak dapat meningkatkan kapasitas dengan memori yang cukup untuk menangani beban kerja dan fitur intensif memori apa pun yang diaktifkan.
- Ubah beban kerja pada instance yang dimaksud. Misalnya, Anda dapat menambahkan instance pembaca ke cluster untuk menyebarkan beban baca di lebih banyak instance.
- Sesuaikan kueri aplikasi Anda sehingga mereka menggunakan lebih sedikit sumber daya.
- Coba gunakan instance yang disediakan yang lebih besar dari maksimum yang NCUs tersedia dalam Neptunus Tanpa Server, untuk melihat apakah itu lebih cocok untuk memori dan persyaratan CPU dari beban kerja.

Menggunakan Amazon Neptune Tanpa Server

Anda dapat membuat cluster DB Neptunus baru sebagai kluster tanpa server, atau dalam beberapa kasus Anda dapat mengonversi cluster DB yang ada untuk menggunakan tanpa server. Anda juga dapat mengonversi instans DB dalam cluster DB tanpa server ke dan dari instans tanpa server. Anda hanya dapat menggunakan Neptunus Tanpa Server di salah Wilayah AWS satu tempat yang didukung, dengan beberapa batasan lainnya (lihat). Kendala Amazon Neptunus Tanpa Server

Anda juga dapat menggunakan tumpukan CloudFormation Neptunus untuk membuat cluster DB Neptunus Tanpa Server.

Membuat cluster DB baru yang menggunakan Serverless

Untuk membuat cluster DB Neptunus yang menggunakan tanpa server, Anda dapat melakukannya dengan cara yang sama seperti yang Anda lakukan untuk membuat cluster yang disediakan. Konsol Manajemen AWS Perbedaannya adalah bahwa di bawah ukuran instans DB, Anda perlu mengatur kelas instance DB ke tanpa server. Ketika Anda melakukannya, Anda kemudian perlu mengatur rentang kapasitas tanpa server untuk cluster.

Anda juga dapat membuat cluster DB tanpa server menggunakan perintah AWS CLI with seperti ini (di Windows, ganti '\' dengan '^'):

```
aws neptune create-db-cluster \
  --region (an Wilayah AWS region that supports serverless) \
  --db-cluster-identifier (ID for the new serverless DB cluster) \
  --engine neptune \
  --engine-version (optional: 1.2.0.1 or above) \
  --serverless-v2-scaling-configuration "MinCapacity=1.0, MaxCapacity=128.0"
```

Anda juga dapat menentukan `serverless-v2-scaling-configuration` parameter seperti ini:

```
  --serverless-v2-scaling-configuration '{"MinCapacity":1.0, "MaxCapacity":128.0}'
```

Anda kemudian dapat menjalankan `describe-db-clusters` perintah untuk `ServerlessV2ScalingConfiguration` atribut, yang akan mengembalikan pengaturan rentang kapasitas yang Anda tentukan:

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": (the specified minimum number of NCUs),
```

```
    "MaxCapacity": (the specified maximum number of NCUs)
}
```

Mengonversi cluster atau instance DB yang ada ke Tanpa Server

Jika Anda memiliki cluster DB Neptunus yang menggunakan engine versi 1.2.0.1 atau lebih tinggi, Anda dapat mengonversinya menjadi tanpa server. Proses ini memang menimbulkan beberapa downtime.

Langkah pertama adalah menambahkan rentang kapasitas ke cluster yang ada. Anda dapat melakukannya menggunakan Konsol Manajemen AWS, atau dengan menggunakan AWS CLI perintah seperti ini (pada Windows, ganti '\' dengan '^'):

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (your DB cluster ID) \
  --serverless-v2-scaling-configuration \
      MinCapacity=(minimum number of NCUs, such as  2.0), \
      MaxCapacity=(maximum number of NCUs, such as 24.0)
```

Langkah selanjutnya adalah membuat instance DB tanpa server baru untuk menggantikan instance primer yang ada (penulis) di cluster. Sekali lagi, Anda dapat melakukan ini dan semua langkah selanjutnya menggunakan salah satu Konsol Manajemen AWS atau AWS CLI. Dalam kedua kasus, tentukan kelas instance DB sebagai tanpa server. AWS CLI Perintah akan terlihat seperti ini (di Windows, ganti '\' dengan '^'):

```
aws neptune create-db-instance \
  --db-instance-identifier (an instance ID for the new writer instance) \
  --db-cluster-identifier (ID of the DB cluster) \
  --db-instance-class db.serverless \
  --engine neptune
```

Ketika instance penulis baru telah tersedia, lakukan failover untuk menjadikannya instance penulis untuk cluster:

```
aws neptune failover-db-cluster \
  --db-cluster-identifier (ID of the DB cluster) \
  --target-db-instance-identifier (instance ID of the new serverless instance)
```

Selanjutnya, hapus contoh penulis lama:

```
aws neptune delete-db-instance \
  --db-instance-identifier (instance ID of the old writer instance) \
  --skip-final-snapshot
```

Terakhir, lakukan hal yang sama untuk membuat instance tanpa server baru untuk menggantikan setiap instance pembaca yang ada yang ingin Anda ubah menjadi instance tanpa server, dan hapus instance yang disediakan yang ada (tidak diperlukan failover untuk instance pembaca).

Memodifikasi rentang kapasitas cluster DB tanpa server yang ada

Anda dapat mengubah rentang kapasitas cluster DB Neptunus Tanpa Server menggunakan AWS CLI seperti ini (di Windows, ganti '\' dengan '^'):

```
aws neptune modify-db-cluster \
  --region (an AWS region that supports serverless) \
  --db-cluster-identifier (ID of the serverless DB cluster) \
  --apply-immediately \
  --serverless-v2-scaling-configuration MinCapacity=4.0, MaxCapacity=32
```

Mengubah rentang kapasitas menyebabkan perubahan pada nilai default dari beberapa parameter konfigurasi. Neptunus dapat segera menerapkan beberapa default baru tersebut, tetapi beberapa perubahan parameter dinamis hanya berlaku setelah reboot. Status `pending-reboot` menunjukkan bahwa Anda memerlukan reboot untuk menerapkan beberapa perubahan parameter.

Mengubah instans DB Tanpa Server menjadi disediakan

Yang perlu Anda lakukan untuk mengonversi instance Neptunus Tanpa Server ke instance yang disediakan adalah mengubah kelas instancenya menjadi salah satu kelas instance yang disediakan. Lihat Memodifikasi instans DB Neptune (dan Menerapkan Segera).

Mengkonfigurasi klien Gremlin untuk Tanpa Server

Saat menggunakan WebSocket klien Gremlin dengan Neptunus Tanpa Server, Anda perlu mengonfigurasi interval detak jantung klien dengan tepat untuk mempertahankan koneksi yang stabil selama peristiwa penskalaan. Untuk petunjuk konfigurasi terperinci untuk klien Java, Go, JavaScript / Node.js, dan Python, lihat. Konfigurasi Detak Jantung untuk Neptunus Tanpa Server

Memantau kapasitas tanpa server dengan Amazon CloudWatch

Anda dapat menggunakannya CloudWatch untuk memantau kapasitas dan pemanfaatan instance tanpa server Neptunus di cluster DB Anda. Ada dua CloudWatch metrik yang memungkinkan Anda melacak kapasitas tanpa server saat ini baik di tingkat cluster maupun di tingkat instans:

- **`ServerlessDatabaseCapacity`**— Sebagai metrik tingkat instance, `ServerlessDatabaseCapacity` laporkan kapasitas instans saat ini, di. NCUs Sebagai metrik tingkat cluster, ia melaporkan rata-rata semua `ServerlessDatabaseCapacity` nilai dari semua instance DB di cluster.
- **`NCUUtilization`**Metrik ini melaporkan persentase kapasitas yang mungkin digunakan. Ini dihitung sebagai arus `ServerlessDatabaseCapacity` (baik pada tingkat instans atau di tingkat cluster) dibagi dengan pengaturan kapasitas maksimum untuk cluster DB.

  Jika metrik ini mendekati 100% pada tingkat cluster, yang berarti bahwa cluster telah berskala setinggi mungkin, pertimbangkan untuk meningkatkan pengaturan kapasitas maksimum.

  Jika mendekati 100% untuk instance pembaca sementara instance penulis tidak mendekati kapasitas maksimum, pertimbangkan untuk menambahkan lebih banyak instance pembaca untuk mendistribusikan beban kerja baca.

Perhatikan bahwa `FreeableMemory` metrik `CPUUtilization` dan memiliki arti yang sedikit berbeda untuk instance tanpa server daripada instance yang disediakan. Dalam konteks tanpa server, `CPUUtilization` adalah persentase yang dihitung sebagai jumlah CPU yang saat ini digunakan dibagi dengan jumlah CPU yang akan tersedia pada kapasitas maksimum. Demikian pula, `FreeableMemory` melaporkan jumlah memori yang dapat dibebaskan yang akan tersedia jika sebuah instance berada pada kapasitas maksimum.

Contoh berikut menunjukkan bagaimana menggunakan AWS CLI on Linux untuk mengambil nilai kapasitas minimum, maksimum, dan rata-rata untuk instans DB tertentu, diukur setiap 10 menit selama satu jam. Perintah `date` Linux menentukan waktu mulai dan akhir secara relatif terhadap tanggal dan waktu saat ini. `sort_by`Fungsi dalam `--query` parameter mengurutkan hasil secara kronologis berdasarkan bidang: `Timestamp`

```
aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '1 hour ago')" \
  --end-time "$(date -d 'now')" \
```

```
  --period 600 \
  --namespace "AWS/Neptune"
  --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=(instance ID) \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
  --output table
```

Menggunakan Amazon Neptunus dengan database global

Basis data global Amazon Neptunus mencakup Wilayah AWS beberapa, memungkinkan pembacaan global latensi rendah dan memberikan pemulihan cepat dalam kasus yang jarang terjadi di mana pemadaman mempengaruhi keseluruhan. Wilayah AWS

Database global Neptunus terdiri dari cluster DB primer di satu wilayah, dan hingga lima cluster DB sekunder di berbagai wilayah.

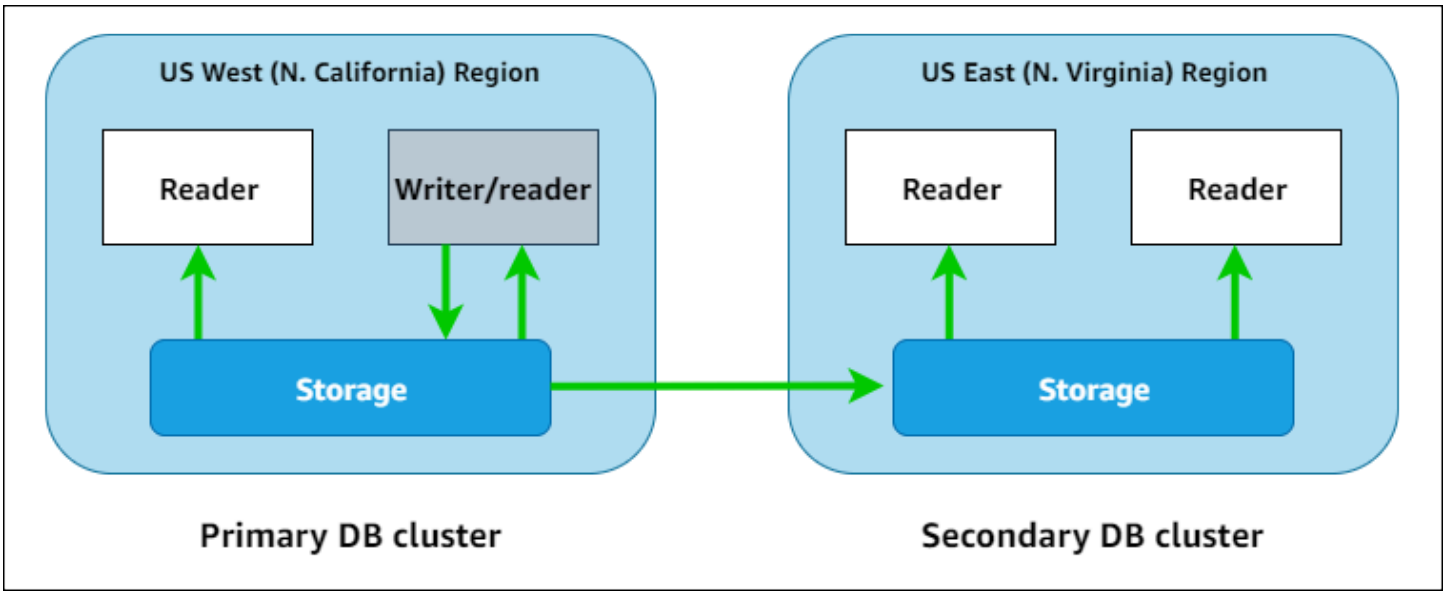
Menulis hanya dapat terjadi di wilayah primer. Wilayah sekunder hanya mendukung pembacaan. Setiap wilayah sekunder dapat memiliki hingga 16 instance pembaca.

Database global di Amazon Neptunus

Dengan menggunakan database global Neptunus, Anda dapat menjalankan aplikasi terdistribusi secara global pada satu database yang mencakup beberapa. Wilayah AWS

Database global Neptunus terdiri dari satu cluster DB di Wilayah AWS primer tempat data ditulis, dan hingga lima cluster DB read-only di sekunder. Wilayah AWS Saat Anda melakukan operasi tulis pada cluster DB primer, Neptunus mereplikasi data tertulis ke semua cluster DB sekunder menggunakan infrastruktur khusus, dengan latensi biasanya di bawah satu detik.

Diagram berikut menunjukkan contoh database global yang mencakup dua Wilayah AWS:



Anda dapat menskalakan setiap klaster sekunder secara independen untuk menangani beban kerja hanya-baca dengan menambahkan satu atau beberapa instance replika baca.

Untuk melakukan operasi tulis, Anda harus terhubung ke titik akhir cluster DB dari cluster DB primer. Hanya cluster primer yang dapat melakukan operasi penulisan. Kemudian, seperti yang ditunjukkan pada diagram di atas, replikasi dilakukan oleh volume penyimpanan cluster, bukan mesin database.

Database global Neptunus dirancang untuk aplikasi dengan jejak di seluruh dunia. Cluster DB sekunder hanya-baca mendukung operasi baca yang lebih dekat dengan pengguna aplikasi.

Database global Neptunus mendukung dua pendekatan berbeda untuk failover:

- Untuk memulihkan dari pemadaman di wilayah primer, gunakan detach-and-promote proses manual yang tidak direncanakan, di mana Anda melepaskan satu cluster sekunder, mengubahnya menjadi cluster mandiri, dan kemudian mempromosikannya menjadi cluster primer baru.
- Untuk prosedur operasional yang direncanakan seperti pemeliharaan, gunakan failover terencana terkelola, di mana Anda memindahkan klaster utama ke salah satu wilayah sekundernya tanpa kehilangan data.

Keuntungan menggunakan database global di Amazon Neptunus

Menggunakan database global, memiliki keuntungan sebagai berikut:

- Bacaan global dengan latensi lokal — Jika Anda memiliki kantor di seluruh dunia, database global memungkinkan kantor Anda di wilayah sekunder mengakses data di wilayah mereka sendiri dengan latensi lokal.
- Cluster DB Neptunus sekunder yang dapat diskalakan — Anda dapat menskalakan cluster sekunder dengan menambahkan instance DB replika baca. Karena cluster sekunder hanya baca, mereka masing-masing dapat mendukung hingga 16 replika baca daripada batas 15 yang biasa.
- Replikasi cepat ke cluster DB sekunder - Replikasi dari cluster DB primer ke sekunder cepat, dengan latensi biasanya di bawah satu detik, dengan sedikit dampak kinerja pada cluster DB primer. Karena replikasi dilakukan pada tingkat penyimpanan, sumber daya instans DB sepenuhnya tersedia untuk beban kerja baca dan tulis aplikasi.
- Pemulihan dari pemadaman di seluruh wilayah - Cluster DB sekunder memungkinkan Anda memindahkan cluster utama ke wilayah baru lebih cepat, dengan RTO yang lebih rendah dan kehilangan data yang lebih sedikit (RPO yang lebih rendah) daripada solusi replikasi tradisional.

Keterbatasan database global di Amazon Neptunus

Batasan berikut saat ini berlaku untuk database global:

- Database global Neptunus hanya tersedia dalam hal berikut: Wilayah AWS
  - US East (N. Virginia): `us-east-1`
  - AS Timur (Ohio): `us-east-2`
  - US West (N. California): `us-west-1`
  - US West (Oregon): `us-west-2`
  - Kanada Barat (Calgary): `ca-west-1`
  - Eropa (Spanyol): `eu-south-2`
  - Eropa (Irlandia): `eu-west-1`
  - Eropa (London): `eu-west-2`
  - Eropa (Frankfurt): `eu-central-1`
  - Asia Pacific (Tokyo): `ap-northeast-1`
  - Asia Pasifik (Osaka): `ap-northeast-3`
  - Asia Pacific (Singapore): `ap-southeast-1`
  - Asia Pasifik (Jakarta): `ap-southeast-3`
  - Asia Pasifik (Melbourne): `ap-southeast-4`
  - Asia Pasifik (Malaysia): `ap-southeast-5`
  - Israel (Tel Aviv): `il-central-1`
- Database global Neptunus tidak `db.t3.medium` mendukung atau jenis instance. `db.t4g.medium`
- Database global Neptunus tidak mendukung auto-scaling untuk cluster DB sekunder.
- Anda tidak dapat menerapkan grup parameter kustom ke cluster database global saat Anda melakukan upgrade versi utama dari database global tersebut. Sebagai gantinya, buat grup parameter kustom Anda di setiap wilayah cluster global dan kemudian terapkan secara manual ke cluster regional setelah upgrade.
- Anda tidak dapat menghentikan atau memulai cluster DB dalam database global satu per satu.
- Instans baca-replika di cluster DB sekunder dapat dimulai ulang dalam keadaan tertentu, termasuk peningkatan yang direncanakan selama jendela pemeliharaan Anda. Jika instance penulis klaster primer dimulai ulang atau gagal, semua instance di wilayah sekunder juga dimulai ulang. Cluster

sekunder kemudian tidak tersedia sampai semua instance-nya kembali sinkron dengan instance penulis cluster DB primer.

Menyiapkan database global Amazon Neptunus

Anda dapat membuat database global Neptunus dengan salah satu cara berikut:

- Buat database global dengan semua cluster dan instance DB baru.
- Buat database global menggunakan cluster DB Neptunus yang ada sebagai cluster utama.

Topik

- Persyaratan konfigurasi untuk database global di Amazon Neptunus
- Menggunakan AWS CLI untuk membuat database global di Amazon Neptunus
- Mengubah cluster DB yang ada menjadi database global
- Menambahkan wilayah database global sekunder ke wilayah utama di Amazon Neptunus
- Menghubungkan ke database global Neptunus

Persyaratan konfigurasi untuk database global di Amazon Neptunus

Database global Neptunus mencakup setidaknya dua. Wilayah AWS Primer Wilayah AWS berisi cluster DB Neptunus yang memiliki satu contoh penulis. Satu hingga lima sekunder Wilayah AWS masing-masing berisi cluster DB Neptunus hanya-baca yang seluruhnya terdiri dari contoh replika baca. Setidaknya satu sekunder Wilayah AWS diperlukan.

Cluster DB Neptunus yang membentuk database global memiliki persyaratan spesifik berikut:

- Persyaratan kelas instans DB — Database global membutuhkan `r5` atau kelas instans `r6g` DB yang dioptimalkan untuk beban kerja intensif memori, seperti jenis instance`db.r5.large`.
- Wilayah AWS persyaratan — Database global membutuhkan cluster DB Neptunus primer dalam Wilayah AWS satu, dan setidaknya satu cluster DB Neptunus sekunder di wilayah yang berbeda. Anda dapat membuat hingga lima cluster DB Neptunus hanya-baca sekunder, dan masing-masing harus berada di wilayah yang berbeda. Dengan kata lain, tidak ada dua cluster DB Neptunus dalam database global Neptunus yang bisa sama. Wilayah AWS
- Persyaratan versi mesin — Versi mesin Neptunus yang digunakan oleh semua cluster DB dalam database global harus sama, dan harus lebih besar dari atau sama dengan. `1.2.0.0` Jika Anda

tidak menentukan versi mesin saat membuat database global baru atau cluster atau instance, versi mesin terbaru akan digunakan.

> **Important**
>
> Meskipun grup parameter cluster DB dapat dikonfigurasi secara independen untuk setiap cluster DB dalam database global, yang terbaik adalah menjaga pengaturan tetap konsisten di seluruh cluster untuk menghindari perubahan perilaku yang tidak terduga jika cluster sekunder harus dipromosikan ke primer. Misalnya, gunakan pengaturan yang sama untuk indeks objek, aliran, dan sebagainya di semua cluster DB.

Menggunakan AWS CLI untuk membuat database global di Amazon Neptunus

> **Note**
>
> Contoh di bagian ini mengikuti konvensi UNIX menggunakan garis miring terbalik (\) sebagai karakter line-extender. Untuk Windows, ganti backslach dengan tanda sisipan (). ^

Untuk membuat database global menggunakan AWS CLI

1. Mulailah dengan membuat database global kosong menggunakan `create-global-cluster` AWS CLI perintah (yang membungkus CreateGlobalCluster API). Tentukan nama Wilayah AWS yang ingin Anda jadikan primer, atur Neptunus sebagai mesin database, dan secara opsional, tentukan versi mesin yang ingin Anda gunakan (ini harus versi 1.2.0.0 atau lebih tinggi):

   ```
   aws neptune create-global-cluster
     --region (primary region, such as us-east-1) \
     --global-cluster-identifier (ID for the global database) \
     --engine neptune \
     --engine-version (engine version; this is optional)
   ```

2. Diperlukan beberapa menit agar database global tersedia, jadi sebelum melanjutkan ke langkah berikutnya, gunakan perintah `describe-global-clusters` CLI (yang membungkus DescribeGlobalClusters API) untuk memeriksa apakah database global tersedia:

```
aws neptune describe-global-clusters \
  --region (primary region) \
  --global-cluster-identifier (global database ID)
```

3. Setelah database global Neptunus tersedia, Anda dapat membuat cluster DB Neptunus baru untuk menjadi cluster utamanya:

```
aws neptune create-db-cluster \
  --region (primary region) \
  --db-cluster-identifier (ID for the primary DB cluster) \
  --engine neptune \
  --engine-version (engine version; must be >= 1.2.0.0) \
  --global-cluster-identifier (global database ID)
```

4. Gunakan `describe-db-clusters` AWS CLI perintah untuk mengonfirmasi bahwa cluster DB baru siap bagi Anda untuk menambahkan instance DB utamanya:

```
aws neptune describe-db-clusters \
  --region (primary region) \
  --db-cluster-identifier (primary DB cluster ID)
```

   Ketika respons muncul`"Status": "available"`, lanjutkan ke langkah berikutnya.

5. Buat instance DB primer untuk cluster primer menggunakan `create-db-instance` AWS CLI perintah. Anda harus menggunakan salah satu jenis yang dioptimalkan memori `r5` atau `r6g` contoh, seperti. `db.r5.large`

```
aws neptune create-db-instance \
  --region (primary region) \
  --db-cluster-identifier (primary cluster ID) \
  --db-instance-class (instance class) \
  --db-instance-identifier (ID for the DB instance) \
  --engine neptune \
  --engine-version (optional: engine version)
```

> **Note**
>
> Jika Anda berencana untuk menambahkan data ke cluster DB primer baru menggunakan pemuat massal Neptunus, lakukan sebelum menambahkan wilayah sekunder. Ini lebih

cepat dan lebih hemat biaya daripada melakukan beban massal setelah database global sepenuhnya disiapkan.

Sekarang tambahkan satu atau lebih wilayah sekunder ke database global baru, seperti yang dijelaskan dalamMenambahkan wilayah sekunder menggunakan AWS CLI.

Mengubah cluster DB yang ada menjadi database global

Untuk mengubah cluster DB yang ada menjadi database global, gunakan `create-global-cluster` AWS CLI perintah untuk membuat database global baru Wilayah AWS sama dengan cluster DB yang ada, dan atur `--source-db-cluster-identifier` parameternya ke Amazon Resource Name (ARN) dari cluster yang ada di sana:

```
aws neptune create-global-cluster \
  --region (region where the existing cluster is located) \
  --global-cluster-identifier (provide an ID for the new global database) \
  --source-db-cluster-identifier (the ARN of the existing DB cluster) \
  --engine neptune \
  --engine-version (engine version; this is optional)
```

Sekarang tambahkan satu atau lebih wilayah sekunder ke database global baru, seperti yang dijelaskan dalamMenambahkan wilayah sekunder menggunakan AWS CLI.

Gunakan cluster DB yang dipulihkan dari snapshot sebagai cluster utama

Anda dapat mengubah cluster DB yang dipulihkan dari snapshot menjadi database global Neptunus. Setelah restore selesai, putar cluster DB yang dibuatnya menjadi cluster utama database global baru seperti yang dijelaskan di atas.

Menambahkan wilayah database global sekunder ke wilayah utama di Amazon Neptunus

Database global Neptunus membutuhkan setidaknya satu cluster DB Neptunus sekunder yang berbeda dari cluster DB primer. Wilayah AWS Anda dapat melampirkan hingga lima cluster DB sekunder ke cluster DB primer.

Setiap cluster DB sekunder yang Anda tambahkan mengurangi satu jumlah maksimum instance read-replica yang dapat Anda miliki di cluster utama. Misalnya, jika ada 4 cluster sekunder, maka

jumlah maksimum instance baca-replika yang dapat Anda miliki di cluster utama adalah. `15 - 4 = 11` Ini berarti bahwa jika Anda sudah memiliki 14 instance pembaca di cluster DB primer dan satu cluster sekunder, Anda tidak akan dapat menambahkan cluster sekunder lainnya.

Menggunakan AWS CLI untuk menambahkan wilayah sekunder ke database global di Neptunus

Untuk menambahkan sekunder Wilayah AWS ke database global Neptunus menggunakan AWS CLI

1. Gunakan `create-db-cluster` AWS CLI perintah untuk membuat cluster DB baru di wilayah yang berbeda dari cluster utama Anda, dan atur `--global-cluster-identifier` parameternya untuk menentukan ID database global:

```
aws neptune create-db-cluster \
  --region (the secondary region) \
  --db-cluster-identifier (ID for the new secondary DB cluster) \
  --global-cluster-identifier (global database ID)
  --engine neptune \
  --engine-version (optional: engine version)
```

2. Gunakan `describe-db-clusters` AWS CLI perintah untuk mengonfirmasi bahwa cluster DB baru siap bagi Anda untuk menambahkan instance DB utamanya:

```
aws neptune describe-db-clusters \
  --region (primary region) \
  --db-cluster-identifier (primary DB cluster ID)
```

   Ketika respons muncul`"Status": "available"`, lanjutkan ke langkah berikutnya.

3. Buat instance DB primer untuk cluster primer menggunakan `create-db-instance` AWS CLI perintah, menggunakan tipe instance di kelas `r5` atau `r6g` instance:

```
aws neptune create-db-instance \
  --region (secondary region) \
  --db-cluster-identifier (secondary cluster ID) \
  --db-instance-class (instance class) \
  --db-instance-identifier (ID for the DB instance) \
  --engine neptune \
  --engine-version (optional: engine version)
```

> **Note**
>
> Jika Anda tidak bermaksud untuk melayani sejumlah besar permintaan baca di wilayah sekunder, dan terutama berkaitan dengan menjaga data Anda didukung dengan andal di sana, Anda dapat membuat cluster DB sekunder tanpa instans DB. Ini menghemat uang, karena Anda hanya membayar untuk penyimpanan cluster sekunder, yang Neptunus akan tetap sinkron dengan penyimpanan di cluster DB primer.

Menghubungkan ke database global Neptunus

Bagaimana Anda terhubung ke database global Neptunus tergantung pada apakah Anda perlu menulis atau membacanya:

- Untuk permintaan atau kueri hanya-baca, sambungkan ke titik akhir pembaca untuk cluster Neptunus di Anda. Wilayah AWS
- Untuk menjalankan kueri mutasi, sambungkan ke titik akhir cluster untuk cluster DB primer, yang mungkin berbeda Wilayah AWS dari aplikasi Anda.

Mengelola basis data global Amazon Neptunus

Dengan pengecualian proses failover terencana terkelola, Anda melakukan sebagian besar operasi manajemen pada kluster individu yang membentuk database global Neptunus. Proses failover terencana yang dikelola hanya tersedia untuk database global Neptunus, bukan untuk kluster DB Neptunus individu. Untuk mempelajari selengkapnya, lihat Melakukan kegagalan terencana terkelola untuk database global Neptunus.

Untuk memulihkan database global Neptunus dari pemadaman yang tidak direncanakan di wilayah utamanya, lihat. Detach-and-promote database global Neptunus dalam kasus pemadaman yang tidak direncanakan

Meskipun Anda dapat mengonfigurasi grup parameter cluster DB secara independen untuk setiap cluster Neptunus dalam database global, yang terbaik adalah menjaga pengaturan tetap konsisten di antara semua cluster untuk menghindari perubahan perilaku yang tidak terduga jika cluster sekunder dipromosikan menjadi yang utama. Misalnya, gunakan pengaturan yang sama untuk indeks objek, aliran, dan sebagainya di semua cluster DB.

Menghapus cluster DB dari database global Neptunus

Ada beberapa alasan Anda mungkin ingin menghapus cluster DB dari database global. Contoh:

- Jika klaster primer menjadi terdegradasi atau terisolasi, Anda dapat menghapusnya dari database global dan menjadi klaster yang disediakan mandiri yang dapat digunakan untuk membuat database global baru (lihat). Detach-and-promote database global Neptunus dalam kasus pemadaman yang tidak direncanakan
- Jika Anda ingin menghapus database global, pertama-tama Anda harus menghapus (melepaskan) semua cluster terkait darinya, meninggalkan primer untuk yang terakhir (lihat. Menghapus database global Neptunus

Anda dapat menggunakan perintah remove-from-global-clusterCLI (yang membungkus RemoveFromGlobalCluster API) untuk melepaskan cluster DB Neptunus dari database global:

```
aws neptune remove-from-global-cluster \
  --region (region of the cluster to remove) \
  --global-cluster-identifier (global database ID) \
  --db-cluster-identifier (ARN of the cluster to remove)
```

Cluster DB yang terpisah kemudian menjadi cluster DB mandiri.

Menghapus database global Neptunus

Anda tidak dapat menghapus database global dan klaster terkait dalam satu langkah. Sebagai gantinya, Anda harus menghapus komponennya satu per satu:

1. Lepaskan semua cluster DB sekunder dari database global, seperti yang dijelaskan dalam. Menghapus cluster Jika mau, Anda sekarang dapat menghapusnya satu per satu.
2. Lepaskan cluster DB primer dari database global.
3. Hapus semua instance DB read-replica dari cluster utama.
4. Hapus instans DB primer (penulis) dari cluster utama. Jika Anda melakukan ini di konsol, itu juga menghapus cluster DB.
5. Hapus database global itu sendiri. Untuk melakukan ini menggunakan AWS CLI, gunakan perintah delete-global-clusterCLI (yang membungkus DeleteGlobalCluster API), sebagai berikut:

   ```
   aws neptune delete-global-cluster \
   ```

```
  --region (region of the DB cluster to delete) \
  --global-cluster-identifier (global database ID)
```

Memodifikasi database global Neptunus

Grup parameter cluster DB dapat dikonfigurasi secara independen untuk setiap cluster DB Neptunus dalam database global, tetapi yang terbaik adalah menjaga pengaturan tetap konsisten di seluruh cluster untuk menghindari perubahan perilaku yang tidak terduga jika cluster sekunder harus dipromosikan ke primer.

Anda dapat memodifikasi pengaturan database global itu sendiri menggunakan perintah modify-global-clusterCLI (yang membungkus API). ModifyGlobalCluster Misalnya, Anda dapat mengubah pengenal database global dan pada saat yang sama mematikan perlindungan penghapusan seperti ini:

```
aws neptune modify-global-cluster \
  --region (region of the DB cluster to modify) \
  --global-cluster-identifier (current global database ID) \
  --new-global-cluster-identifier (new global database ID to assign) \
  --no-deletion-protection
```

Melakukan pemulihan bencana Amazon Neptunus

Database global Neptunus menyediakan kemampuan failover yang lebih komprehensif daripada cluster DB Neptunus mandiri. Dengan menggunakan database global, Anda dapat merencanakan dan memulihkan diri dari bencana dengan cukup cepat. Pemulihan bencana umumnya dinilai menggunakan evaluasi tujuan waktu pemulihan (RTO) dan tujuan titik pemulihan (RPO):

- Recovery-time Objective (RTO) — Ini adalah seberapa cepat sistem kembali ke keadaan kerja setelah bencana. Dengan kata lain, RTO mengukur waktu henti operasional. Untuk database global Neptunus, RTO bisa dalam urutan menit.
- Recovery-point Objective (RPO) — Jumlah waktu di mana data hilang. Untuk database global Neptunus, RPO biasanya diukur dalam hitungan detik (lihat). Melakukan kegagalan terencana terkelola untuk database global Neptunus

Untuk database global Neptunus, ada dua pendekatan berbeda untuk failover:

- Detach-and-promote (pemulihan tidak terencana manual) - Untuk memulihkan dari pemadaman yang tidak direncanakan atau untuk melakukan pengujian pemulihan bencana (pengujian DR), lakukan lintas wilayah detach-and-promote pada salah satu cluster DB sekunder dalam database global. RTO untuk proses manual ini tergantung pada seberapa cepat Anda dapat melakukan tugas yang tercantum dalam Lepaskan dan promosikan. RPO biasanya beberapa detik, tetapi ini tergantung pada lag replikasi penyimpanan di seluruh jaringan pada saat kegagalan.
- Failover terencana terkelola — Pendekatan ini ditujukan untuk pemeliharaan operasional dan prosedur operasional terencana lainnya seperti merelokasi cluster DB primer dari database global ke salah satu wilayah sekunder. Karena proses ini menyinkronkan cluster DB sekunder dengan primer sebelum membuat perubahan lain, RPO secara efektif 0 (yaitu, tidak ada kehilangan data). Lihat Melakukan kegagalan terencana terkelola untuk database global Neptunus.

Detach-and-promote database global Neptunus dalam kasus pemadaman yang tidak direncanakan

Dalam situasi yang sangat langka di mana basis data global Neptunus Anda mengalami pemadaman tak terduga di primer Wilayah AWS, cluster DB Neptunus utama Anda dan simpul penulisnya menjadi tidak tersedia, dan replikasi antara cluster primer dan sekunder berhenti. Untuk meminimalkan downtime yang dihasilkan (RTO) dan kehilangan data (RPO), lakukan lintas wilayah dengan cepat detach-and-promote untuk merekonstruksi database global.

> **Tip**
>
> Sebaiknya pahami proses ini sebelum menggunakannya, dan memiliki rencana untuk melanjutkan dengan cepat pada tanda pertama masalah di seluruh wilayah.
>
> - Gunakan Amazon CloudWatch secara teratur untuk melacak waktu jeda untuk cluster sekunder sehingga Anda dapat mengidentifikasi wilayah sekunder dengan waktu jeda terkecil jika Anda perlu gagal.
> - Pastikan untuk menguji rencana Anda untuk memeriksa apakah prosedur Anda lengkap dan akurat.
> - Gunakan lingkungan simulasi untuk memastikan staf Anda terlatih dan siap untuk melakukan failover DR dengan cepat jika diperlukan.

Untuk gagal ke cluster sekunder setelah pemadaman yang tidak direncanakan di wilayah primer

1. Berhenti mengeluarkan kueri mutasi dan operasi penulisan lainnya di cluster DB primer.
2. Identifikasi cluster DB di sekunder Wilayah AWS untuk digunakan sebagai cluster DB primer baru dari database global. Jika database global memiliki dua atau lebih sekunder Wilayah AWS, pilih cluster sekunder yang memiliki waktu jeda terkecil.
3. Lepaskan cluster DB sekunder yang Anda pilih dari database global Neptunus.

   Menghapus cluster DB sekunder dari database global Neptunus segera menghentikan replikasi data dari primer ke sekunder itu dan mempromosikannya ke cluster DB mandiri dengan kemampuan penuh. read/write Cluster sekunder lainnya dalam database global akan tetap tersedia dan dapat menerima panggilan baca dari aplikasi Anda.

   Sebelum membuat ulang database global Neptunus, Anda juga harus melepaskan cluster sekunder lainnya untuk menghindari inkonsistensi data di antara cluster (lihat). Menghapus cluster
4. Konfigurasikan ulang aplikasi Anda untuk mengirim semua operasi penulisan ke cluster DB Neptunus mandiri yang Anda pilih untuk menjadi cluster utama baru, menggunakan titik akhir barunya. Jika Anda menerima nama default saat membuat database global Neptunus, Anda dapat mengubah titik akhir dengan menghapus `-ro` string titik akhir dari cluster di aplikasi Anda.

   Misalnya, titik akhir cluster sekunder `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com` menjadi `my-global.cluster-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com` ketika cluster itu terlepas dari database global.

   Cluster DB Neptunus ini menjadi cluster utama dari database global Neptunus baru ketika Anda mulai menambahkan wilayah ke dalamnya pada langkah berikutnya.
5. Tambahkan Wilayah AWS ke cluster DB. Saat Anda melakukannya, proses replikasi dari klaster primer ke klaster sekunder akan dimulai. Lihat Menambahkan wilayah database global sekunder ke wilayah utama di Amazon Neptunus.
6. Tambahkan lebih banyak Wilayah AWS sesuai kebutuhan untuk membuat ulang topologi yang diperlukan untuk mendukung aplikasi Anda.

Pastikan bahwa penulisan aplikasi dikirim ke cluster DB Neptunus yang benar sebelum, selama, dan setelah melakukan perubahan ini. Melakukan hal ini menghindari inkonsistensi data di antara cluster DB di database global Neptunus (ini dikenal sebagai masalah otak terpisah).

Melakukan kegagalan terencana terkelola untuk database global Neptunus

Failover terencana terkelola memungkinkan Anda memindahkan cluster utama database global Neptunus Anda ke yang berbeda kapan pun Anda mau. Wilayah AWS Beberapa organisasi akan ingin merotasi lokasi cluster utama mereka secara teratur.

> **Note**
>
> Proses failover terencana terkelola yang dijelaskan di sini dimaksudkan untuk digunakan pada database global Neptunus yang sehat. Untuk pulih dari pemadaman yang tidak direncanakan atau untuk melakukan pengujian pemulihan bencana (DR), ikuti proses pelepasan dan promosikan sebagai gantinya.

Selama failover terencana terkelola, klaster utama Anda gagal ke wilayah sekunder pilihan Anda sementara topologi replikasi database global Anda yang ada dipertahankan. Sebelum proses failover terencana terkelola dimulai, database global menyinkronkan semua cluster sekunder dengan cluster utamanya. Setelah memastikan bahwa semua cluster disinkronkan, failover terencana yang dikelola dimulai. Cluster DB di wilayah primer menjadi hanya-baca, dan cluster sekunder yang dipilih mempromosikan salah satu instance hanya-baca ke status penulis penuh, sehingga memungkinkan cluster untuk mengambil peran klaster primer. Karena semua cluster sekunder disinkronkan dengan primer pada awal proses, primer baru melanjutkan operasi untuk database global tanpa kehilangan data apa pun. Basis data hanya tidak tersedia untuk waktu yang singkat sementara cluster sekunder primer dan terpilih mengasumsikan peran baru mereka.

Untuk mengoptimalkan ketersediaan aplikasi, lakukan failover selama jam nonpeak, pada saat penulisan ke cluster DB primer minimal. Juga, ambil langkah-langkah berikut sebelum memulai failover:

- Bawa aplikasi offline sedapat mungkin untuk mengurangi penulisan ke cluster utama.
- Periksa waktu jeda untuk semua cluster DB Neptunus sekunder dalam database global dan pilih yang sekunder dengan waktu jeda keseluruhan paling sedikit untuk menjadi yang utama. Gunakan Amazon CloudWatch untuk melihat `NeptuneGlobalDBProgressLag` metrik untuk semua sekunder. Metrik ini memberi tahu Anda seberapa jauh sekunder berada di belakang cluster DB primer, dalam milidetik. Nilainya berbanding lurus dengan waktu yang dibutuhkan Neptunus untuk menyelesaikan failover. Dengan kata lain, semakin besar nilai lag, semakin lama pemadaman failover, jadi pilih yang sekunder dengan lag paling sedikit. Untuk informasi selengkapnya, lihat Metrik Neptunus CloudWatch.

Selama failover terencana terkelola, cluster DB sekunder yang dipilih dipromosikan ke peran barunya sebagai primer tetapi tidak mewarisi konfigurasi lengkap cluster DB primer. Ketidakcocokan dalam konfigurasi dapat menyebabkan masalah performa, inkompatibilitas beban kerja, dan perilaku anomali lainnya. Untuk menghindari masalah seperti itu, selesaikan jenis perbedaan konfigurasi berikut antara cluster database global sebelum failover:

- Konfigurasikan parameter di primer baru agar sesuai dengan primer saat ini.
- Konfigurasikan alat pemantauan, opsi, dan alarm — Konfigurasikan cluster DB yang akan menjadi primer baru dengan kemampuan logging, alarm, dan sebagainya yang dimiliki primer saat ini.
- Konfigurasikan integrasi dengan AWS layanan lain — Jika database global Neptunus Anda terintegrasi AWS dengan layanan, AWS Identity and Access Management seperti (IAM), Amazon S3, atau AWS Lambda, pastikan ini dikonfigurasi sesuai kebutuhan untuk berintegrasi dengan cluster DB primer yang baru.

Ketika proses failover selesai dan cluster DB yang dipromosikan siap menangani operasi penulisan untuk database global, pastikan untuk mengubah aplikasi Anda untuk menggunakan titik akhir baru untuk primer baru.

Menggunakan AWS CLI untuk memulai failover terencana terkelola

Gunakan perintah failover-global-clusterCLI (yang membungkus FailoverGlobalCluster API) untuk gagal atas database global Neptunus Anda:

```
aws neptune failover-global-cluster \
  --region (the region where the primary cluster is located) \
  --global-cluster-identifier (global database ID) \
  --target-db-cluster-identifier (the ARN of the secondary DB cluster to promote)
```

Memantau basis data global Amazon Neptunus Anda menggunakan metrik CloudWatch

Neptunus mendukung metrik CloudWatch berikut yang dapat Anda gunakan untuk memantau database global Neptunus:

- **`GlobalDbDataTransferBytes`**— Jumlah byte data redo log yang ditransfer dari primer Wilayah AWS ke sekunder Wilayah AWS dalam database global Neptunus.

- **`GlobalDbReplicatedWriteIO`**— Jumlah I/O operasi tulis direplikasi dari primer Wilayah AWS dalam database global ke volume cluster di sekunder Wilayah AWS.

  Perhitungan penagihan untuk setiap cluster DB dalam database global Neptunus menggunakan metrik untuk memperhitungkan penulisan `VolumeWriteIOPS` yang dilakukan dalam cluster itu. Untuk cluster DB primer, perhitungan penagihan digunakan `NeptuneGlobalDbReplicatedWriteIO` untuk memperhitungkan replikasi lintas wilayah ke cluster DB sekunder.
- **`GlobalDbProgressLag`**— Jumlah milidetik yang cluster sekunder berada di belakang cluster utama untuk transaksi pengguna dan transaksi sistem.

Metrik	Dimensi	Diterbitkan di	Unit
`GlobalDbDataTransferBytes`	SourceRegion, DBCluster Pengidentifikasi	Sekunder	Byte
`GlobalDbReplicatedWriteIO`	SourceRegion, DBCluster Pengidentifikasi	Sekunder	Hitungan
`GlobalDbProgressLag`	DBClusterPengidentifikasi, Secondary Region : di DBCluster Pengidentifikasi Utama, SourceRegion : di Sekunder	Primer, Sekunder	Milidetik

Ikhtisar fitur Amazon Neptunus

> **Note**
>
> Bagian ini tidak mencakup penggunaan bahasa kueri yang dapat Anda gunakan untuk mengakses data dalam grafik Neptunus.
> Untuk informasi tentang cara menyambung ke klaster DB Neptune yang sedang berjalan dengan Gremlin, lihat Mengakses grafik Neptune dengan Gremlin.
> Untuk informasi tentang cara menghubungkan ke cluster DB Neptunus yang sedang berjalan dengan OpenCypher, lihat. Mengakses Grafik Neptunus dengan OpenCypher
> Untuk informasi tentang cara menyambung ke klaster DB Neptune yang sedang berjalan dengan SPARQL, lihat Mengakses grafik Neptune dengan SPARQL.

Bagian ini memberikan ikhtisar fitur Neptunus tertentu, termasuk:

- Kepatuhan Neptune dengan standar bahasa kueri.
- Model data grafik Neptune.
- Penjelasan semantik transaksi Neptunus.
- Pengantar tentang klaster dan instans Neptune.
- Penyimpanan, keandalan, dan ketersediaan Neptune.
- Penjelasan tentang titik akhir Neptune.
- Menggunakan mode laboratorium Neptune untuk mengaktifkan fitur eksperimental.
- Deskripsi mesin DFE Neptunus.
- Konektivitas JDBC Neptunus.
- Daftar rilis mesin Neptune dan cara memperbarui mesin Anda.

Catatan tentang Kepatuhan Standar Amazon Neptune

Amazon Neptune mematuhi standar yang berlaku dalam menerapkan bahasa kueri grafik Gremlin dan SPARQL dalam banyak kasus.

Bagian ini menggambarkan standar serta area-area di mana Neptune meluas atau menyimpang dari standar dan area tersebut.

Topik

- Kepatuhan standar Gremlin di Amazon Neptune
- Kepatuhan standar SPARQL di Amazon Neptunus
- Kepatuhan spesifikasi OpenCypher di Amazon Neptunus

Kepatuhan standar Gremlin di Amazon Neptune

Bagian berikut memberikan gambaran umum tentang implementasi Neptunus Gremlin dan bagaimana hal itu berbeda dari implementasi Apache. TinkerPop

Neptunus mengimplementasikan beberapa langkah Gremlin secara native di mesinnya, dan menggunakan implementasi TinkerPop Apache Gremlin untuk memproses yang lain (lihat). Dukungan langkah Gremlin asli di Amazon Neptune

> **Note**
>
> Untuk beberapa contoh konkret dari perbedaan implementasi ini yang ditunjukkan di Konsol Gremlin dan Amazon Neptune, lihat bagian the section called “Menggunakan Gremlin” dari Quick Start.

Topik

- Standar yang Berlaku untuk Gremlin
- Variabel dan parameter dalam skrip
- TinkerPop enumerasi
- Kode Java
- Properti pada elemen
- Eksekusi skrip

- Sesi
- Transaksi
- Vertex dan tepi IDs
- Disediakan pengguna IDs
- Properti Vertex IDs
- Kardinalitas sifat simpul
- Memperbarui properti vertex
- Label
- Karakter melarikan diri
- Keterbatasan Groovy
- Serialisasi
- Langkah-langkah Lambda
- Metode Gremlin yang tidak didukung
- Langkah-langkah Gremlin yang tidak didukung
- Fitur grafik Gremlin di Neptunus

Standar yang Berlaku untuk Gremlin

- Bahasa Gremlin didefinisikan oleh TinkerPop Dokumentasi Apache dan TinkerPop implementasi Apache dari Gremlin bukan oleh spesifikasi formal.
- Untuk format numerik, Gremlin mengikuti standar IEEE 754 (IEEE 754-2019 - Standar IEEE untuk Floating-Point Arithmetic. Untuk informasi selengkapnya, lihat juga halaman Wikipedia IEEE 754).

Variabel dan parameter dalam skrip

Jika variabel pra-terikat bersangkutan, objek traversal g adalah Pre-bound di Neptunus, dan objek tidak didukung. `graph`

Meskipun Neptunus tidak mendukung variabel Gremlin atau parameterisasi dalam skrip, Anda mungkin sering menemukan skrip sampel untuk Server Gremlin di Internet yang berisi deklarasi variabel, seperti:

```
String query = "x = 1; g.V(x)";
```

```
List<Result> results = client.submit(query).all().get();
```

Ada juga banyak contoh yang menggunakan parameterisasi (atau binding) saat mengirimkan kueri, seperti:

```
Map<String,Object> params = new HashMap<>();
params.put("x",1);
String query = "g.V(x)";
List<Result> results = client.submit(query).all().get();
```

Contoh parameter biasanya dikaitkan dengan peringatan tentang hukuman kinerja karena tidak membuat parameter bila memungkinkan. Ada banyak sekali contoh seperti itu TinkerPop yang mungkin Anda temui, dan semuanya terdengar cukup meyakinkan tentang perlunya membuat parameter.

Namun, baik fitur deklarasi variabel dan fitur parameterisasi (bersama dengan peringatan) hanya berlaku untuk Server TinkerPop Gremlin saat menggunakan. `GremlinGroovyScriptEngine` Mereka tidak berlaku ketika Gremlin Server menggunakan tata bahasa `gremlin-language` ANTLR Gremlin untuk mengurai kueri. Tata bahasa ANTLR tidak mendukung deklarasi variabel atau parameterisasi, jadi saat menggunakan ANTLR, Anda tidak perlu khawatir gagal membuat parameter. Karena tata bahasa ANTLR adalah komponen yang lebih baru TinkerPop, konten lama yang mungkin Anda temui di Internet umumnya tidak mencerminkan perbedaan ini.

Neptunus menggunakan tata bahasa ANTLR di mesin pemrosesan kueri daripada`GremlinGroovyScriptEngine`, sehingga tidak mendukung variabel atau parameterisasi atau properti. `bindings` Akibatnya, masalah yang terkait dengan kegagalan parameterisasi tidak berlaku di Neptunus. Menggunakan Neptunus, sangat aman hanya untuk mengirimkan kueri apa adanya di mana seseorang biasanya akan membuat parameter. Akibatnya, contoh sebelumnya dapat disederhanakan tanpa penalti kinerja sebagai berikut:

```
String query = "g.V(1)";
List<Result> results = client.submit(query).all().get();
```

TinkerPop enumerasi

Neptune tidak mendukung nama kelas yang memenuhi syarat untuk nilai pencacahan. Misalnya, Anda harus menggunakan `single` dan bukan `org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single` dalam permintaan Groovy Anda.

Jenis pencacahan ditentukan oleh jenis parameter.

Tabel berikut menunjukkan nilai enumerasi yang diizinkan dan nama yang TinkerPop sepenuhnya memenuhi syarat terkait.

Nilai yang Diizinkan	Kelas
`id`, `key`, `label`, `value`	org.apache.tinkerpop.gremlin.structure.t
`T.id`, `T.key`, `T.label`, `T.value`	org.apache.tinkerpop.gremlin.structure.t
`set`, `single`	org.apache.tinkerpop.gremlin.structure. VertexProperty.Kardinalitas
`asc`, `desc`, `shuffle`	org.apache.tinkerpop.gremlin.process.traversal.order
`Order.asc` , `Order.desc` , `Order.shuffle`	org.apache.tinkerpop.gremlin.process.traversal.order
`global`, `local`	org.apache.tinkerpop.gremlin.process.traversal.scope
`Scope.global` , `Scope.local`	org.apache.tinkerpop.gremlin.process.traversal.scope
`all`, `first`, `last`, `mixed`	org.apache.tinkerpop.gremlin.process.traversal.pop
`normSack`	org.apache.tinkerpop.gremlin.process.traversal. SackFunctions.Penghalang
`addAll`, `and`, `assign`, `div`, `max`, `min`, `minus`, `mult`, `or`, `sum`, `sumLong`	org.apache.tinkerpop.gremlin.process.traversal.operator
`keys`, `values`	org.apache.tinkerpop.gremlin.structure.column
`BOTH`, `IN`, `OUT`	org.apache.tinkerpop.gremlin.structure.direction
`any`, `none`	org.apache.tinkerpop.gremlin.process.traversal.step. TraversalOptionParent.Pilih

Kode Java

Neptunus tidak mendukung panggilan ke metode yang ditentukan oleh panggilan perpustakaan Java atau Java arbitrer selain Gremlin yang didukung. APIs Misalnya, `java.lang.*`, `Date()`, dan `g.V().tryNext().orElseGet()` tidak diperbolehkan.

Properti pada elemen

Neptunus tidak mendukung bendera `materializeProperties` yang diperkenalkan di 3.7.0 untuk mengembalikan TinkerPop properti pada elemen. Akibatnya, Neptunus masih hanya akan mengembalikan simpul atau tepi sebagai referensi hanya dengan dan. `id` `label`

Eksekusi skrip

Semua kueri harus dimulai dengan g, objek traversal.

Dalam pengiriman kueri String, beberapa traversals dapat dikeluarkan terpisah dengan titik koma (`;`) atau karakter baris baru (\n). Untuk dieksekusi, setiap pernyataan selain yang terakhir harus diakhiri dengan langkah `.iterate()`. Hanya data dari traversal akhir yang dikembalikan. Perhatikan bahwa ini tidak berlaku untuk pengiriman ByteCode kueri GLV.

Sesi

Sesi di Neptune dibatasi hanya durasi 10 menit. Lihat Sesi berbasis skrip Gremlin dan Referensi TinkerPop Sesi untuk informasi lebih lanjut.

Transaksi

Neptune membuka transaksi baru pada awal setiap traversal Gremlin dan menutup transaksi setelah berhasil menyelesaikan traversal. Transaksi di-rollback ketika ada kesalahan.

Beberapa pernyataan dipisahkan dengan titik koma (`;`) atau karakter baris baru (\n) disertakan dalam satu transaksi. Setiap pernyataan selain yang terakhir harus diakhiri dengan langkah `next()` yang akan dieksekusi. Hanya data dari traversal akhir yang dikembalikan.

Logika transaksi manual menggunakan `tx.commit()` dan `tx.rollback()` tidak didukung.

> **Important**
>
> Ini hanya berlaku untuk metode di mana Anda mengirim kueri Gremlin sebagai string teks (lihatTransaksi Gremlin).

Vertex dan tepi IDs

Neptunus Gremlin Vertex dan Edge harus bertipe. IDs `String` String ID ini mendukung karakter Unicode, dan ukurannya tidak boleh melebihi 55 MB.

Disediakan pengguna IDs didukung, tetapi opsional dalam penggunaan normal. Jika Anda tidak memberikan ID saat menambahkan simpul atau tepi, Neptunus menghasilkan UUID dan mengubahnya menjadi string, dalam bentuk seperti ini:. `"48af8178-50ce-971a-fc41-8c9a954cea62"` Ini UUIDs tidak sesuai dengan standar RFC, jadi jika Anda membutuhkan standar, UUIDs Anda harus membuatnya secara eksternal dan menyediakannya saat Anda menambahkan simpul atau tepi.

> **Note**
>
> Perintah `Load` Neptunus mengharuskan Anda IDs menyediakan, menggunakan bidang ~id dalam format CSV Neptunus.

Disediakan pengguna IDs

Disediakan pengguna IDs diizinkan di Neptunus Gremlin dengan ketentuan berikut.

- Disediakan IDs adalah opsional.
- Hanya vertex dan edge yang didukung.
- Hanya tipe `String` yang didukung.

Untuk membuat vertex baru dengan ID kustom, gunakan langkah `property` dengan Kata Kunci `id`: `g.addV().property(id, 'customid')`.

> **Note**
>
> Jangan menaruh tanda kutip di sekitar Kata Kunci `id`. Ini mengacu pada`T.id`.

Semua simpul IDs harus unik, dan semua tepi IDs harus unik. Namun, Neptune tidak mengizinkan vertex dan edge untuk memiliki ID yang sama.

Jika Anda mencoba untuk membuat sebuah vertex baru menggunakan `g.addV()` dan sebuah vertex dengan ID sudah ada, operasi gagal. Pengecualian untuk ini adalah jika Anda menentukan label baru

untuk vertex tersebut, operasi berhasil tetapi menambahkan label baru dan properti tambahan yang ditentukan ke vertex yang ada. Tidak ada yang ditimpa. Sebuah vertex baru tidak dibuat. ID vertex tidak berubah dan tetap unik.

Misalnya, perintah konsol Gremlin berikut berhasil:

```
gremlin> g.addV('label1').property(id, 'customid')
gremlin> g.addV('label2').property(id, 'customid')
gremlin> g.V('customid').label()
==>label1::label2
```

Properti Vertex IDs

Properti Vertex IDs dihasilkan secara otomatis dan dapat muncul sebagai angka positif atau negatif ketika ditanyakan.

Kardinalitas sifat simpul

Neptune mendukung kardinalitas rangkaian dan kardinalitas tunggal. Jika tidak ditentukan, kardinalitas rangkaian dipilih. Ini berarti bahwa jika Anda menetapkan nilai properti, ia menambahkan nilai baru ke properti, tetapi hanya jika itu belum muncul di set nilai. Ini adalah nilai pencacahan Gremlin Set.

`List` tidak didukung. Untuk informasi lebih lanjut tentang kardinalitas properti, lihat topik Vertex di Gremlin. JavaDoc

Memperbarui properti vertex

Untuk memperbarui nilai properti tanpa menambahkan nilai tambahan untuk set nilai, tentukan kardinalitas `single` dalam langkah `property`.

```
g.V('exampleid01').property(single, 'age', 25)
```

Ini akan menghapus semua nilai yang ada untuk properti tersebut.

Label

Neptune mendukung beberapa label untuk sebuah vertex. Ketika Anda membuat label, Anda dapat menentukan beberapa label dengan memisahkannya dengan `::`. Sebagai contoh, `g.addV("Label1::Label2::Label3")` menambahkan sebuah vertex dengan tiga label yang

berbeda. Langkah `hasLabel` cocok dengan vertex ini dengan salah satu dari tiga label tersebut: `hasLabel("Label1")`, `hasLabel("Label2")`, dan `hasLabel("Label3")`.

> **Important**
>
> Pembatas `::` dicadangkan untuk penggunaan ini saja. Anda tidak dapat menentukan beberapa label di langkah `hasLabel`. Misalnya, `hasLabel("Label1::Label2")` tidak cocok dengan apa pun.

Karakter melarikan diri

Neptune menyelesaikan semua karakter escape seperti yang dijelaskan dalambagian Meng-Escape Karakter Khusus dari dokumentasi bahasa Apache Groovy.

Keterbatasan Groovy

Neptune tidak mendukung perintah Groovy yang tidak dimulai dengan `g`. Ini termasuk matematika (misalnya, `1+1`), panggilan sistem (misalnya, `System.nanoTime()`), dan definisi variabel (misalnya, `1+1`).

> **Important**
>
> Neptune tidak mendukung nama kelas yang memenuhi syarat.
> Misalnya, Anda harus menggunakan `single` dan bukan `org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single` dalam permintaan Groovy Anda.

Serialisasi

Neptune mendukung serialisasi berikut berdasarkan jenis MIME yang diminta.

Neptunus mengekspos semua serializer TinkerPop yang melakukannya, dengan dukungan untuk berbagai versi dan konfigurasi GraphSon dan. GraphBinary Meskipun ada banyak pilihan yang ada, panduan untuk digunakan sangat mudah:

- Jika Anda menggunakan TinkerPop driver Apache, pilih default untuk driver tanpa menentukan satu secara eksplisit. Kecuali Anda memiliki alasan yang sangat spesifik untuk melakukannya,

Anda mungkin tidak perlu menentukan serializer dalam inisialisasi driver Anda. Secara umum, default yang digunakan oleh driver adalah`application/vnd.graphbinary-v1.0`.

- Jika Anda terhubung ke Neptunus melalui HTTP, prioritaskan penggunaan `application/vnd.gremlin-v3.0+json;types=false` sebagai tipe tertanam dalam versi alternatif GraphSon 3 membuatnya rumit untuk dikerjakan.
- `application/vnd.graphbinary-v1.0-stringd`Umumnya hanya berguna bila digunakan bersama dengan Gremlin Console karena mengubah semua hasil menjadi representasi string untuk tampilan sederhana.
- Format yang tersisa tetap ada karena alasan lama dan biasanya tidak boleh digunakan dengan driver tanpa alasan yang jelas.

Jenis MIME	Serialisasi	Konfigurasi
`application/vnd.gremlin-v1.0+json`	`GraphSONMessageSerializerV1`	`ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV1]`
`application/vnd.gremlin-v1.0+json;types=false`	`GraphSONUntypedMessageSerializerV1`	`ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV1]`
`application/vnd.gremlin-v2.0+json`	`GraphSONMessageSerializerV2`	`ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV2]`
`application/vnd.gremlin-v2.0+json;types=false`	`GraphSONUntypedMessageSerializerV2`	`ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV2]`

application/vnd.gremlin-v3.0+json	GraphSONMessageSerializerV3	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV3]
application/vnd.gremlin-v3.0+json;types=false	GraphSONUntypedMessageSerializerV3	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV3]
application/json	GraphSONUntypedMessageSerializerV3	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV1]
application/vnd.graphbinary-v1.0	GraphBinaryMessageSerializerV1	
application/vnd.graphbinary-v1.0-stringd	GraphBinaryMessageSerializerV1	serializeResultToString: true
application/vnd.gremlin-v1.0+json	GraphSONMessageSerializerGremlinV1	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV1]
application/vnd.gremlin-v2.0+json	GraphSONMessageSerializerV2 (hanya bekerja dengan WebSockets)	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV2]

`application/vnd.gremlin-v3.0+json`	`GraphSONMessageSerializerV3`	
`application/json`	`GraphSONMessageSerializerV3`	`ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV3]`
`application/vnd.graphbinary-v1.0`	`GraphBinaryMessageSerializerV1`	

> **Note**
>
> Tabel serializer yang ditampilkan di sini mengacu pada penamaan pada 3.7.0. TinkerPop Jika Anda ingin tahu lebih banyak tentang perubahan ini, silakan lihat dokumentasi TinkerPop pemutakhiran. Dukungan serialisasi Gryo tidak digunakan lagi di 3.4.3 dan secara resmi dihapus di 3.6.0. Jika Anda secara eksplisit menggunakan Gryo atau pada versi driver yang menggunakannya secara default, maka Anda harus beralih ke GraphBinary atau meningkatkan driver Anda.

Langkah-langkah Lambda

Neptune tidak mendukung Lambda Steps.

Metode Gremlin yang tidak didukung

Neptune tidak mendukung metode Gremlin berikut:

- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.progra`
- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.sideEf`
- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.from(o`
- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.to(org`

Misalnya, traversal berikut tidak diperbolehkan:
`g.V().addE('something').from(__.V().next()).to(__.V().next())`.

> **Important**
>
> Ini hanya berlaku untuk metode di mana Anda mengirim kueri Gremlin sebagai string teks.

Langkah-langkah Gremlin yang tidak didukung

Neptune tidak mensupport langkah-langkah Gremlin berikut:

- Langkah Gremlin io () hanya didukung sebagian di Neptunus. Ini dapat digunakan dalam konteks baca, seperti dalam`g.io(`*`url`*`).read()`, tetapi tidak untuk menulis.

Fitur grafik Gremlin di Neptunus

Implementasi Gremlin Neptune tidak mengekspos`graph`objek. Tabel berikut mencantumkan fitur Gremlin dan menunjukkan apakah Neptunus mendukungnya atau tidak.

Dukungan Neptunus untuk fitur **`graph`**

Fitur grafik Neptunus, jika didukung, sama seperti yang akan dikembalikan oleh perintah. `graph.features()`

Fitur grafik	Diaktifkan?
`Transactions`	true
`ThreadedTransactions`	SALAH
`Computer`	SALAH
`Persistence`	true
`ConcurrentAccess`	true

Dukungan Neptunus untuk fitur variabel

Fitur variabel	Diaktifkan?
`Variables`	false

SerializableValues	false
UniformListValues	false
BooleanArrayValues	false
DoubleArrayValues	false
IntegerArrayValues	false
StringArrayValues	false
BooleanValues	false
ByteValues	false
DoubleValues	false
FloatValues	false
IntegerValues	false
LongValues	false
MapValues	false
MixedListValues	false
StringValues	false
ByteArrayValues	false
FloatArrayValues	false
LongArrayValues	false

Dukungan Neptunus untuk fitur vertex

Fitur Vertex	Diaktifkan?
MetaProperties	false

DuplicateMultiProperties	SALAH
AddVertices	true
RemoveVertices	true
MultiProperties	true
UserSuppliedIds	true
AddProperty	true
RemoveProperty	true
NumericIds	SALAH
StringIds	BETUL
UuidIds	SALAH
CustomIds	false
AnyIds	false

Dukungan Neptunus untuk fitur properti vertex

Fitur properti Vertex	Diaktifkan?
UserSuppliedIds	SALAH
AddProperty	true
RemoveProperty	true
NumericIds	true
StringIds	true
UuidIds	SALAH
CustomIds	false

AnyIds	SALAH
Properties	BETUL
SerializableValues	SALAH
UniformListValues	false
BooleanArrayValues	false
DoubleArrayValues	false
IntegerArrayValues	false
StringArrayValues	SALAH
BooleanValues	true
ByteValues	true
DoubleValues	true
FloatValues	true
IntegerValues	true
LongValues	true
MapValues	SALAH
MixedListValues	SALAH
StringValues	BETUL
ByteArrayValues	SALAH
FloatArrayValues	false
LongArrayValues	false

Dukungan Neptunus untuk fitur tepi

Fitur tepi	Diaktifkan?
`AddEdges`	true
`RemoveEdges`	true
`UserSuppliedIds`	true
`AddProperty`	true
`RemoveProperty`	true
`NumericIds`	SALAH
`StringIds`	BETUL
`UuidIds`	SALAH
`CustomIds`	false
`AnyIds`	false

Dukungan Neptunus untuk fitur properti edge

Fitur properti tepi	Diaktifkan?
`Properties`	true
`SerializableValues`	SALAH
`UniformListValues`	false
`BooleanArrayValues`	false
`DoubleArrayValues`	false
`IntegerArrayValues`	false
`StringArrayValues`	SALAH

`BooleanValues`	true
`ByteValues`	true
`DoubleValues`	true
`FloatValues`	true
`IntegerValues`	true
`LongValues`	true
`MapValues`	SALAH
`MixedListValues`	SALAH
`StringValues`	BETUL
`ByteArrayValues`	SALAH
`FloatArrayValues`	false
`LongArrayValues`	false

Kepatuhan standar SPARQL di Amazon Neptunus

Setelah mencantumkan standar SPARQL yang berlaku, bagian berikut memberikan rincian spesifik tentang bagaimana implementasi SPARQL Neptunus meluas atau menyimpang dari standar tersebut.

Topik

- Standar Berlaku untuk SPARQL
- Default Namespace Awalan di Neptune SPARQL
- Grafik Standar SPARQL dan Grafik Bernama
- Fungsi XPath Konstruktor SPARQL Didukung oleh Neptunus
- Basis IRI default untuk kueri dan pembaruan
- XSD: Nilai DateTime di Neptune
- Neptune Penanganan Nilai Titik Terapung Khusus

- Batasan Neptune Nilai Sewenang-Panjang
- Neptune Memperpanjang Perbandingan Sama di SPARQL
- Penanganan Out-of-Range Literal di Neptunus SPARQL

Amazon Neptune mematuhi standar berikut dalam menerapkan bahasa kueri grafik SPARQL.

Standar Berlaku untuk SPARQL

- SPARQL didefinisikan oleh rekomendasi W3C Bahasa Kueri SPARQL 1.1 21 Maret 2013.
- Protokol pembaruan dan bahasa kueri SPARQL didefinisikan oleh spesifikasi W3C Pembaruan SPARQL 1.1.
- Untuk format numerik, SPARQL mengikuti spesifikasi Bahasa Definisi Skema XML W3C (XSD) 1.1 Bagian 2: Tipe data, yang konsisten dengan spesifikasi IEEE 754 (IEEE 754-2019 - Standar IEEE untuk Floating-Point Arithmetic. Untuk informasi selengkapnya, lihat juga halaman Wikipedia IEEE 754). Namun, fitur yang diperkenalkan setelah versi `IEEE 754-1985` tidak termasuk dalam spesifikasi.

Default Namespace Awalan di Neptune SPARQL

Neptune mendefinisikan awalan berikut secara default untuk digunakan dalam kueri SPARQL. Untuk informasi selengkapnya, lihat Nama Prefiks dalam spesifikasi SPARQL.

- `rdf – http://www.w3.org/1999/02/22-rdf-syntax-ns#`
- `rdfs – http://www.w3.org/2000/01/rdf-schema#`
- `owl – http://www.w3.org/2002/07/owl#`
- `xsd – http://www.w3.org/2001/XMLSchema#`

Grafik Standar SPARQL dan Grafik Bernama

Amazon Neptune mengasosiasikan setiap tripel dengan grafik bernama. Grafik default ditetapkan sebagai gabungan dari semua grafik bernama.

Grafik Default untuk Query

Jika Anda mengirimkan kueri SPARQL tanpa secara eksplisit menentukan grafik melalui kata kunci `GRAPH` atau konstruksi seperti `FROM NAMED`, Neptune selalu menganggap semua tripel dalam

instans DB Anda. Misalnya, kueri berikut mengembalikan semua tripel dari titik akhir Neptune SPARQL:

```
SELECT * WHERE { ?s ?p ?o }
```

Tripel yang muncul di lebih dari satu grafik dikembalikan hanya sekali.

Untuk informasi tentang spesifikasi grafik default, lihat bagian Set Data RDF dari spesifikasi Bahasa Kueri SPARQL 1.1.

Menentukan grafik bernama untuk Loading, menyisipkan, atau update

Jika Anda tidak menentukan grafik bernama ketika memuat, memasukkan, atau memperbarui tripel, Neptune menggunakan grafik bernama fallback yang ditentukan oleh URI, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Saat Anda mengeluarkan permintaan `Load` Neptune menggunakan format berbasis triple, Anda dapat menentukan grafik bernama untuk menggunakan untuk semua triple dengan menggunakan parameter `parserConfiguration: namedGraphUri`. Untuk informasi tentang menggunakan sintaks perintah `Load`, lihat the section called “Perintah Loader”.

> **Important**
>
> Jika Anda tidak menggunakan parameter ini, dan Anda tidak menentukan grafik bernama, URI mundur digunakan: `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Grafik bernama mundur ini juga digunakan jika Anda memuat triple melalui `SPARQL UPDATE` tanpa secara eksplisit memberikan target grafik bernama.

Anda dapat menggunakan format berbasis paha depan N-Quads untuk menentukan grafik bernama untuk setiap triple dalam database.

> **Note**
>
> Menggunakan N-Quads memungkinkan Anda untuk meninggalkan grafik bernama kosong. Dalam kasus ini, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph` digunakan.

Anda dapat mengganti grafik bernama default untuk N-Quads menggunakan opsi konfigurasi parser `namedGraphUri`.

Fungsi XPath Konstruktor SPARQL Didukung oleh Neptunus

Standar SPARQL memungkinkan mesin SPARQL untuk mendukung serangkaian fungsi konstruktor yang dapat diperluas. XPath Neptune saat ini mendukung fungsi konstruktor berikut, di mana `xsd` didefinisikan sebagai `http://www.w3.org/2001/XMLSchema#`:

- `xsd:boolean`
- `xsd:integer`
- `xsd:double`
- `xsd:float`
- `xsd:decimal`
- `xsd:long`
- `xsd:unsignedLong`

Basis IRI default untuk kueri dan pembaruan

Karena cluster Neptunus memiliki beberapa titik akhir yang berbeda, menggunakan URL permintaan kueri atau pembaruan sebagai IRI dasar dapat menyebabkan hasil yang tidak terduga saat menyelesaikan relatif. IRIs

Pada rilis mesin 1.2.1.0, Neptunus menggunakan IRI `http://aws.amazon.com/neptune/default/` sebagai basis jika basis eksplisit IRI bukan bagian dari permintaan.

Dalam permintaan berikut, basis IRI adalah bagian dari permintaan:

```
BASE <http://example.org/default/>
INSERT DATA { <node1> <id> "n1" }

BASE <http://example.org/default/>
SELECT * { <node1> ?p ?o }
```

Dan hasilnya adalah:

```
?p                                                    ?o
```

```
http://example.org/default/id                    n1
```

Namun, dalam permintaan ini, tidak ada IRI dasar yang disertakan:

```
INSERT DATA { <node1> <id> "n1" }

SELECT * { <node1> ?p ?o }
```

Dalam hal ini, hasilnya adalah:

```
?p                                               ?o
http://aws.amazon.com/neptune/default/id         n1
```

XSD: Nilai DateTime di Neptune

Untuk alasan kinerja, Neptunus selalu date/time menyimpan nilai sebagai Coordinated Universal Time (UTC). Hal ini membuat perbandingan langsung sangat efisien.

Ini juga berarti bahwa jika Anda memasukkan nilai `dateTime` yang menentukan zona waktu tertentu, Neptune menerjemahkan nilai UTC dan membuang informasi zona waktu tersebut. Kemudian, ketika Anda mengambil nilai `dateTime` nanti, nilai itu dinyatakan dalam UTC, bukan waktu zona waktu asli, dan Anda tidak dapat lagi tahu apa zona waktu aslinya.

Neptune Penanganan Nilai Titik Terapung Khusus

Neptune menangani nilai-nilai floating-point khusus di SPARQL sebagai berikut.

SPARQL NaN Penanganan di Neptune

Di Neptune, SPARQL dapat menerima nilai `NaN` dalam kueri. Tidak ada perbedaan antara nilai `NaN` yang mengeluarkan sinyal dan diam. Neptune memperlakukan semua nilai `NaN` sebagai diam.

Secara semantik, tidak ada perbandingan `NaN` adalah mungkin, karena tidak ada yang lebih besar dari, kurang dari, atau sama dengan `NaN`. Ini berarti bahwa nilai `NaN` di satu sisi perbandingan dalam teori tidak pernah cocokdengan apa pun di sisi lain.

Namun, Spesifikasi XSD benar-benar menganggap dua nilai `NaN` `xsd:double` atau `xsd:float` sebagai sama. Neptune mengikuti ini untuk filter `IN`, untuk operator yang sama dalam ekspresi filter, dan untuk semantik pencocokan tepat (memiliki `NaN` dalam posisi objek pola triple).

SPARQL Penanganan Nilai Tak Terbatas di Neptune

Di Neptune, SPARQL dapat menerima nilai `INF` atau `-INF` dalam kueri. `INF`membandingkan sebagai lebih besar dari nilai numerik lainnya, dan `-INF`membandingkan sebagai kurang dari nilai numerik lainnya.

Dua nilai INF dengan tanda yang cocok dibandingkan sama satu sama lain terlepas dari jenisnya (misalnya, float `-INF` dibandingkan sebagai sama dengan `-INF` ganda).

Tentu saja, tidak ada perbandingan dengan `NaN` adalah mungkin karena tidak ada yang lebih besar dari, kurang dari, atau sama dengan `NaN`.

Penanganan Nol Negatif SPARQL di Neptune

Neptune menormalkan nilai nol negatif ke nol unsigned. Anda dapat menggunakan nilai nol negatif dalam kueri, tetapi mereka tidak direkam seperti itu dalam database, dan mereka membandingkan sebagai sama dengan nol unsigned.

Batasan Neptune Nilai Sewenang-Panjang

Neptune membatasi ukuran penyimpanan XSD integer, floating point, dan nilai desimal dalam SPARQL menjadi 64 bit. Menggunakan nilai yang lebih besar menghasilkan `InvalidNumericDataException` kesalahan.

Neptune Memperpanjang Perbandingan Sama di SPARQL

Standar SPARQL mendefinisikan logika terner untuk ekspresi nilai, di mana ekspresi nilai dapat mengevaluasi `true`, `false`, atau `error`. Semantik default untuk kesetaraan istilah sebagaimana didefinisikan dalam spesifikasi SPARQL 1.1), yang berlaku untuk perbandingan = dan `!`= di kondisi `FILTER`, menghasilkan `error` ketika membandingkan tipe data yang tidak secara eksplisit sebanding dalam tabel operator dalam spesifikasi.

Perilaku ini dapat menyebabkan hasil yang tidak intuitif, seperti dalam contoh berikut.

Data

```
<http://example.com/Server/1> <http://example.com/ip> "127.0.0.1"^^<http://example.com/
datatype/IPAddress>
```

Permintaan 1:

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o = "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

Permintaan 2:

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o != "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

Dengan semantik SPARQL default yang Neptune digunakan sebelum rilis 1.0.2.1, kedua kueri akan mengembalikan hasil kosong. Alasannya adalah bahwa `?o = "127.0.0.2"^^<http://example.com/IPAddress>` ketika dievaluasi untuk `?o := "127.0.0.1"^^<http://example.com/IPAddress>` menghasilkan `error` ketimbang `false` karena tidak ada aturan perbandingan eksplisit yang ditentukan untuk tipe data kustom `<http://example.com/IPAddress>`. Akibatnya, versi dinegasikan dalam kueri kedua juga menghasilkan `error`. Dalam kedua kueri, `error` menyebabkan solusi kandidat untuk disaring.

Dimulai dengan rilis 1.0.2.1, Neptune telah memperpanjang operator ketidaksetaraan SPARQL sesuai dengan spesifikasi. Lihat Bagian SPARQL 1.1 pada perluasan operator, yang memungkinkan mesin untuk menentukan aturan tambahan tentang cara membandingkan seluruh tipe data yang ditetapkan pengguna dan built-in yang tidak dapat dibandingkan.

Menggunakan opsi ini, Neptune sekarang memperlakukan perbandingan dari setiap dua tipe data yang tidak secara eksplisit didefinisikan dalam tabel pemetaan operator sebagai mengevaluasi ke `true` jika nilai-nilai literal dan tipe data yang secara sintaksis sama, dan false sebaliknya. `error` tidak diproduksi dalam kasus apa pun.

Menggunakan semantik baru ini, kueri kedua akan mengembalikan `"127.0.0.1"^^<http://example.com/IPAddress>` alih-alih hasil kosong.

Penanganan Out-of-Range Literal di Neptunus SPARQL

Semantik XSD mendefinisikan setiap jenis numerik dengan ruang nilainya, kecuali untuk `integer` dan `decimal`. Definisi ini membatasi setiap jenis untuk rentang nilai. Sebagai contoh, kisaran dari kisaran `xsd:byte` adalah dari -128 ke +127, inklusif. Setiap nilai di luar kisaran ini dianggap tidak valid.

Jika Anda mencoba menetapkan nilai literal di luar ruang nilai suatu tipe (misalnya, jika Anda mencoba menyetel ke nilai literal 999), Neptunus menerima out-of-range nilai apa adanya, tanpa membulatkan atau memotongnya. `xsd:byte` Tapi itu tidak bertahan sebagai nilai numerik karena jenis yang diberikan tidak bisa mewakilinya.

Artinya, Neptune menerima `"999"^^xsd:byte` meskipun itu adalah nilai di luar kisaran nilai `xsd:byte` yang didefinisikan. Namun, setelah nilai bertahan dalam database, itu hanya dapat digunakan dalam semantik pertandingan tepat, dalam posisi objek dari pola triple. Tidak ada filter rentang yang dapat dijalankan di atasnya karena out-of-range literal tidak diperlakukan sebagai nilai numerik.

Spesifikasi SPARQL 1.1 mendefinisikan operator rentang dalam bentuk `numeric`operator-`numeric`,`string`-operator-`string`,`literal`-operator-`literal`, dan seterusnya. Neptune tidak dapat mengeksekusi operator perbandingan jangkauan seperti `invalid-literal`-operator-`numeric-value`.

Kepatuhan spesifikasi OpenCypher di Amazon Neptunus

Rilis Amazon Neptunus dari OpenCypher umumnya mendukung klausa, operator, ekspresi, fungsi, dan sintaks yang ditentukan oleh spesifikasi OpenCypher saat ini, yang merupakan Referensi Bahasa Kueri Cypher Versi 9. Keterbatasan dan perbedaan dalam dukungan Neptunus untuk OpenCypher disebut di bawah ini.

Amazon Neptunus juga mendukung beberapa fitur di luar cakupan spesifikasi OpenCypher. Lihat Ekstensi OpenCypher di Amazon Neptunus untuk detailnya.

> **Note**
>
> Implementasi Neo4j Cypher saat ini berisi fungsionalitas yang tidak terkandung dalam spesifikasi OpenCypher yang disebutkan di atas. Jika Anda memigrasikan kode Cypher saat ini ke Neptunus, lihat dan untuk informasi lebih lanjut. Kompatibilitas Neptunus dengan Neo4j Menulis ulang kueri Cypher untuk dijalankan di OpenCypher di Neptunus

Support untuk klausa OpenCypher di Neptune

Neptunus mendukung klausa berikut, kecuali seperti yang disebutkan:

- `MATCH`— Didukung, kecuali itu *`shortestPath()`* dan *`allShortestPaths()`* saat ini tidak didukung.

- `OPTIONAL MATCH`
- *`MANDATORY MATCH`*— saat ini tidak didukung di Neptunus. Neptunus, bagaimanapun, mendukung nilai ID kustom dalam kueri. `MATCH`
- `RETURN`— Didukung, kecuali bila digunakan dengan nilai non-statis untuk `SKIP` atau`LIMIT`. Misalnya, berikut ini saat ini tidak berfungsi:

```
MATCH (n)
RETURN n LIMIT toInteger(rand())    // Does NOT work!
```

- `WITH`— Didukung, kecuali bila digunakan dengan nilai non-statis untuk `SKIP` atau`LIMIT`. Misalnya, berikut ini saat ini tidak berfungsi:

```
MATCH (n)
WITH n SKIP toInteger(rand())
WITH count() AS count
RETURN count > 0 AS nonEmpty    // Does NOT work!
```

- `UNWIND`
- `WHERE`
- `ORDER BY`
- `SKIP`
- `LIMIT`
- `CREATE`— Neptunus memungkinkan Anda membuat nilai ID kustom dalam kueri. `CREATE`
- `DELETE`
- `SET`
- `REMOVE`
- `MERGE`- Neptunus mendukung nilai ID kustom dalam kueri. `MERGE`
- *`CALL[YIELD...]`*— saat ini tidak didukung di Neptunus.
- `UNION, UNION ALL`— kueri hanya-baca didukung, tetapi kueri mutasi saat ini tidak didukung.
- `USING`— `USING` didukung dari versi mesin 1.3.2.0. Lihat Petunjuk kueri untuk informasi selengkapnya.

Support untuk operator OpenCypher di Neptune

Neptunus mendukung operator berikut, kecuali seperti yang disebutkan:

Operator umum

- `DISTINCT`
- `.`Operator untuk mengakses properti peta literal bersarang.

Operator matematika

- Operator **+** tambahan.
- `-`Operator pengurangan.
- *****Operator perkalian.
- Operator **/** divisi.
- Operator divisi `%` modulo.
- Operator **^** eksponensial. *`is NOT supported`*

Operator perbandingan

- Operator **=** tambahan.
- Operator **<>** ketidaksetaraan.
- Operator **<** less-than didukung kecuali jika salah satu argumennya adalah Path, List, atau Map.
- Operator **>** yang lebih besar dari didukung kecuali jika salah satu argumennya adalah Path, List, atau Map.
- Operator **<=** less-than-or-equal -to didukung kecuali jika salah satu argumen adalah Path, List, atau Map.
- Operator **>=** greater-than-or-equal -to didukung kecuali jika salah satu argumen adalah Path, List, atau Map.
- `IS NULL`
- `IS NOT NULL`
- `STARTS WITH`didukung jika data yang dicari adalah string.
- `ENDS WITH`didukung jika data yang dicari adalah string.
- `CONTAINS`didukung jika data yang dicari adalah string.

Operator Boolean

- `AND`
- `OR`
- `XOR`
- `NOT`

Operator String

- **+**Operator penggabungan.

Daftar operator

- **+**Operator penggabungan.
- `IN`(memeriksa keberadaan item dalam daftar)

Support untuk ekspresi OpenCypher di Neptune

Neptunus mendukung ekspresi berikut, kecuali seperti yang disebutkan:

- `CASE`
- **`[]`**Ekspresi saat ini tidak didukung di Neptunus untuk mengakses kunci properti yang dihitung secara dinamis dalam node, relasi, atau peta. Misalnya, berikut ini tidak berfungsi:

```
MATCH (n)
WITH [5, n, {key: 'value'}] AS list
RETURN list[1].name
```

Support untuk fungsi OpenCypher di Neptune

Neptunus mendukung fungsi-fungsi berikut, kecuali seperti yang disebutkan:

Fungsi predikat

- `exists()`

Fungsi skalar

- `coalesce()`
- `endNode()`
- `epochmillis()`
- `head()`
- `id()`
- `last()`
- `length()`
- `randomUUID()`
- `properties()`
- `removeKeyFromMap`
- `size()`— metode kelebihan beban saat ini hanya berfungsi untuk ekspresi pola, daftar, dan string
- `startNode()`
- `timestamp()`
- `toBoolean()`
- `toFloat()`
- `toInteger()`
- `type()`

Fungsi agregasi

- `avg()`
- `collect()`
- `count()`
- `max()`
- `min()`
- `percentileDisc()`
- `stDev()`
- `percentileCont()`
- `stDevP()`

- `sum()`

Daftar fungsi

- `join()`(menggabungkan string dalam daftar menjadi satu string)
- `keys()`
- `labels()`
- `nodes()`
- `range()`
- `relationships()`
- `reverse()`
- `tail()`

Fungsi Matematika — Numerik

- `abs()`
- `ceil()`
- `floor()`
- `rand()`
- `round()`
- `sign()`

Fungsi matematika — logaritmik

- `e()`
- `exp()`
- `log()`
- `log10()`
- `sqrt()`

Fungsi matematika — trigonometri

- `acos()`

- `asin()`
- `atan()`
- `atan2()`
- `cos()`
- `cot()`
- `degrees()`
- `pi()`
- `radians()`
- `sin()`
- `tan()`

Fungsi string

- `join()`(menggabungkan string dalam daftar menjadi satu string)
- `left()`
- `lTrim()`
- `replace()`
- `reverse()`
- `right()`
- `rTrim()`
- `split()`
- `substring()`
- `toLower()`
- `toString()`
- `toUpper()`
- `trim()`

Fungsi yang ditentukan pengguna

*`User-defined functions`*Saat ini tidak didukung di Neptunus.

Detail implementasi OpenCypher khusus Neptunus

Bagian berikut menjelaskan cara-cara di mana implementasi Neptunus OpenCypher mungkin berbeda dari atau melampaui spesifikasi OpenCypher.

Evaluasi jalur panjang variabel (VLP) di Neptunus

Variable length path (VLP) evaluasi menemukan jalur antara node dalam grafik. Panjang jalur dapat tidak dibatasi dalam kueri. Untuk mencegah siklus, spesifikasi OpenCypher menetapkan bahwa setiap tepi harus dilalui paling banyak sekali per solusi.

Untuk VLPs, implementasi Neptunus menyimpang dari spesifikasi OpenCypher karena hanya mendukung nilai konstan untuk filter kesetaraan properti. Ambil kueri berikut:

```
MATCH (x)-[:route*1..2 {dist:33, code:x.name}]->(y) return x,y
```

Karena nilai filter kesetaraan `x.name` properti bukan konstanta, kueri ini menghasilkan pesan `UnsupportedOperationException` dengan: `Property predicate over variable-length relationships with non-constant expression is not supported in this release.`

Dukungan temporal dalam implementasi Neptunus OpenCypher (database Neptunus 1.3.1.0 dan di bawahnya)

Neptunus saat ini menyediakan dukungan terbatas untuk fungsi temporal di OpenCypher. Ini mendukung tipe `DateTime` data untuk tipe temporal.

`datetime()`Fungsi ini dapat digunakan untuk mendapatkan tanggal dan waktu UTC saat ini seperti ini:

```
RETURN  datetime() as res
```

Nilai tanggal dan waktu dapat diuraikan dari string dalam '' format T waktu '' tanggal di mana tanggal dan waktu keduanya dinyatakan dalam salah satu formulir yang didukung di bawah ini:

Format tanggal yang didukung

- `yyyy-MM-dd`
- `yyyyMMdd`

- `yyyy-MM`
- `yyyy-DDD`
- `yyyyDDD`
- `yyyy`

Format waktu yang didukung

- `HH:mm:ssZ`
- `HHmmssZ`
- `HH:mm:ssZ`
- `HH:mmZ`
- `HHmmZ`
- `HHZ`
- `HHmmss`
- `HH:mm:ss`
- `HH:mm`
- `HHmm`
- `HH`

Contoh:

```
RETURN datetime('2022-01-01T00:01')      // or another example:
RETURN datetime('2022T0001')
```

Perhatikan bahwa semua date/time nilai di Neptunus OpenCypher disimpan dan diambil sebagai nilai UTC.

Neptunus OpenCypher menggunakan jam, `statement` yang berarti bahwa waktu yang sama digunakan sepanjang durasi kueri. Kueri yang berbeda dalam transaksi yang sama dapat menggunakan waktu instan yang berbeda.

Neptunus tidak mendukung penggunaan fungsi dalam panggilan ke. `datetime()` Misalnya, berikut ini tidak akan berfungsi:

```
CREATE (:n {date:datetime(tostring(2021))})  // ---> NOT ALLOWED!
```

Neptunus memang mendukung fungsi `epochmillis()` yang mengubah a to. `datetime` `epochmillis` Contoh:

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

Neptunus saat ini tidak mendukung fungsi dan operasi lain `DateTime` pada objek, seperti penambahan dan pengurangan.

Dukungan sementara dalam implementasi Neptunus OpenCypher (Neptunus Analytics dan Neptunus Database 1.3.2.0 dan di atasnya)

Fungsionalitas datetime berikut OpenCypher berlaku untuk Neptune Analytics. Atau, Anda dapat menggunakan parameter labmode `DatetimeMillisecond=enabled` untuk mengaktifkan fungsionalitas datetime berikut pada rilis mesin Neptunus versi 1.3.2.0 dan yang lebih baru. Untuk detail selengkapnya tentang penggunaan fungsi ini di labmode, lihatDukungan datetime yang diperpanjang.

- Support untuk milidetik. Datetime literal akan selalu dikembalikan dengan milidetik, bahkan jika milidetik adalah 0. (Perilaku sebelumnya adalah memotong milidetik.)

```
CREATE (:event {time: datetime('2024-04-01T23:59:59Z')})

# Returning the date returns with 000 suffixed representing milliseconds
MATCH(n:event)
RETURN n.time as datetime

{
  "results" : [ {
    "n" : {
      "~id" : "0fe88f7f-a9d9-470a-bbf2-fd6dd5bf1a7d",
      "~entityType" : "node",
      "~labels" : [ "event" ],
      "~properties" : {
        "time" : "2024-04-01T23:59:59.000Z"
      }
    }
  } ]
```

```
}
```

- Support untuk memanggil fungsi datetime () melalui properti yang disimpan atau hasil perantara. Misalnya, kueri berikut tidak dimungkinkan sebelum fitur ini.

  Datetime () atas properti:

```
// Create node with property 'time' stored as string
CREATE (:event {time: '2024-04-01T23:59:59Z'})

// Match and return this property as datetime
MATCH(n:event)
RETURN datetime(n.time) as datetime
```

  Datetime () atas hasil menengah:

```
// Parse datetime from parameter
UNWIND $list as myDate
RETURN datetime(myDate) as d
```

- Sekarang juga dimungkinkan untuk menyimpan kinerja datetime yang dibuat dalam kasus yang disebutkan di atas.

  Menyimpan datetime dari properti string dari satu properti ke properti lainnya:

```
// Create node with property 'time' stored as string
CREATE (:event {time: '2024-04-01T23:59:59Z', name: 'crash'})

// Match and update the same property to datetime type
MATCH(n:event {name: 'crash'})
SET n.time = datetime(n.time)

// Match and update another node's property
MATCH(e:event {name: 'crash'})
MATCH(n:server {name: e.servername})
SET n.time = datetime(e.time)
```

  Batch membuat node dari parameter dengan properti datetime:

```
// Batch create from parameter
UNWIND $list as events
CREATE (n:crash) {time: datetime(events.time)}
```

```
// Parameter value
{
  "x":[
    {"time":"2024-01-01T23:59:29", "name":"crash1"},
    {"time":"2023-01-01T00:00:00Z", "name":"crash2"}
  ]
}
```

- Support untuk subset format ISO8601 datetime yang lebih besar. Lihat di bawah ini.

Format yang didukung

Format nilai datetime adalah [Tanggal] T [Waktu] [Zona Waktu], di mana T adalah pemisah. Jika zona waktu eksplisit tidak disediakan, UTC (Z) dianggap sebagai default.

Zona waktu

Format zona waktu yang didukung adalah:

- +/- HH: mm
- +/-HHmm
- +/- HH

Kehadiran zona waktu dalam string datetime adalah opsional. Jika offset zona waktu adalah 0, Z dapat digunakan sebagai pengganti postfix zona waktu di atas untuk menunjukkan waktu UTC. Rentang zona waktu yang didukung adalah dari - 14:00 hingga + 14:00.

Date

Jika tidak ada zona waktu, atau zona waktunya adalah UTC (Z), format tanggal yang didukung adalah sebagai berikut:

> **Note**
>
> DDD mengacu pada tanggal ordinal, yang mewakili hari dalam setahun dari 001 hingga 365 (366 dalam tahun kabisat). Misalnya, 2024-002 mewakili 2 Januari 2024.

- `yyyy-MM-dd`

- yyyyMMdd
- yyyy-MM
- yyyyMM
- yyyy-DDD
- yyyyDDD
- yyyy

Jika zona waktu selain Z dipilih, format tanggal yang didukung dibatasi sebagai berikut:

- yyyy-MM-dd
- yyyy-DDD
- yyyyDDD

Rentang yang didukung untuk tanggal adalah dari 1400-01-01 hingga 9999-12-31.

Waktu

Jika tidak ada zona waktu, atau zona waktunya adalah UTC (Z), format waktu yang didukung adalah:

- HH:mm:ss.SSS
- HH:mm:ss
- HHmmss.SSS
- HHmmss
- HH:mm
- HHmm
- HH

Jika zona waktu selain Z dipilih, format waktu yang didukung dibatasi sebagai berikut:

- HH:mm:ss
- HH:mm:ss.SSS

Perbedaan semantik bahasa Neptunus OpenCypher

Neptunus mewakili simpul dan IDs hubungan sebagai string daripada bilangan bulat. ID sama dengan ID yang diberikan melalui pemuat data. Jika ada namespace untuk kolom, namespace ditambah ID. Akibatnya, `id` fungsi mengembalikan string bukan integer.

Jenis `INTEGER` data terbatas pada 64 bit. Saat mengonversi nilai floating point atau string yang lebih besar menjadi bilangan bulat menggunakan `TOINTEGER` fungsi, nilai negatif terpotong `LLONG_MIN` dan nilai positif dipotong menjadi. `LLONG_MAX`

Contoh:

```
RETURN TOINTEGER(2^100)
>  9223372036854775807

RETURN TOINTEGER(-1 * 2^100)
>  -9223372036854775808
```

Properti multi-nilai

Meskipun OpenCypher CREATE tidak membuat properti multi-nilai, mereka dapat ada dalam data yang dibuat menggunakan Gremlin (Database Neptunus) atau saat memuat data (Database Neptunus dan Neptunus Analytics). Jika Neptunus OpenCypher menemukan properti multi-nilai, salah satu nilai dipilih secara sewenang-wenang, menciptakan hasil non-deterministik.

Penanganan NaN valuse

Penanganan Neptunus terhadap perbandingan nilai `NaN` properti tidak terdefinisi. Mengandalkan perbandingan tersebut dapat menghasilkan hasil yang tidak terduga atau non-deterministik.

Model Data Grafik Neptune

Unit dasar data grafik Amazon Neptune adalah eleman empat posisi (quad), yang mirip dengan quad Resource Description Framework (RDF). Berikut ini adalah empat posisi quad Neptune:

- `subject (S)`
- `predicate (P)`
- `object (O)`
- `graph (G)`

Setiap quad adalah pernyataan yang membuat penegasan tentang satu sumber daya atau lebih. Sebuah pernyataan dapat menegaskan adanya hubungan antara dua sumber daya, atau dapat melampirkan properti (pasangan nilai-kunci) ke sumber daya. Anda dapat memikirkan nilai predikat quad umumnya sebagai kata kerja pernyataan. Kata kerja tersebut menggambarkan jenis hubungan atau properti yang sedang didefinisikan. Objeknya adalah target hubungan, atau nilai properti. Berikut ini adalah beberapa contohnya:

- Sebuah hubungan antara dua vertex dapat diwakili dengan menyimpan pengidentifikasi vertex sumber di posisi `S`, pengidentifikasi vertex target di posisi `O`, dan label edge di posisi `P`.
- Sebuah properti dapat diwakili dengan menyimpan pengidentifikasi elemen di posisi `S`, kunci properti di posisi `P`, dan nilai properti di posisi `O`.

Posisi grafik `G` digunakan secara berbeda di tumpukan yang berbeda. Untuk data RDF di Neptune, posisi `G` berisi Pengidentifikasi grafik bernama. Untuk grafik properti di Gremlin, digunakan untuk menyimpan nilai ID edge dalam kasus edge. Dalam semua kasus lainnya, diatur default ke nilai tetap.

Satu set pernyataan quad dengan pengidentifikasi sumber daya bersama menciptakan grafik.

Kamus nilai yang dihadapi pengguna

Neptunus tidak menyimpan sebagian besar nilai yang dihadapi pengguna secara langsung di berbagai indeks yang dipertahankannya. Sebaliknya, ia menyimpannya secara terpisah dalam kamus dan menggantinya dalam indeks dengan pengidentifikasi 8-byte.

- Semua nilai yang dihadapi pengguna yang akan masuk`S`,`P`, atau `G` indeks disimpan dalam kamus dengan cara ini.

- Dalam 0 indeks, nilai numerik disimpan langsung dalam indeks (inline). Ini termasuk `date` dan `datetime` nilai (direpresentasikan sebagai milidetik dari zaman).
- Semua nilai yang dihadapi pengguna lainnya yang akan masuk dalam 0 indeks disimpan dalam kamus dan diwakili dalam indeks oleh. IDs

Kamus berisi pemetaan maju dari nilai yang dihadapi pengguna ke 8-byte IDs dalam indeks. `value_to_id`

Ini menyimpan pemetaan terbalik 8-byte IDs ke nilai di salah satu dari dua indeks, tergantung pada ukuran nilai:

- `id_to_value`Indeks memetakan IDs ke nilai yang dihadapi pengguna yang lebih kecil dari 767 byte setelah pengkodean internal.
- `id_to_blob`Indeks memetakan IDs ke nilai yang dihadapi pengguna yang lebih besar.

Bagaimana Pernyataan Diindeks di Neptune

Ketika Anda mengkueri grafik quad, untuk setiap posisi quad, Anda dapat menentukan kendala nilai, atau tidak. Kueri mengembalikan semua quad yang cocok dengan kendala nilai yang Anda tentukan.

Neptunus menggunakan indeks untuk menyelesaikan pola kueri grafik. Indeks ini berada di atas empat komponen utama dari tepi grafik: Subjek (simpul sumber dalam LPG); Predikat (RDF), atau Label Properti atau Tepi (LPG); Objek (simpul target atau nilai properti dalam LPG); dan Grafik (RDF) atau Pengenal Tepi (LPG). Ada 16 (2 ^ 4) kemungkinan pola akses untuk empat posisi komponen quad ini. Anda dapat menanyakan semua 16 pola secara efisien tanpa harus memindai dan memfilter dengan menggunakan enam indeks. Setiap indeks pernyataan quad menggunakan kunci yang terdiri dari empat nilai posisi bersambung dalam urutan yang berbeda. Salah satu kemungkinan kombinasi indeks pernyataan quad yang akan mencakup semua 16 jalur akses adalah:

```
     Access Pattern                                          Index key order
-----------------------------------------------------    ---------------
 1.  ????  (No constraints; returns every quad)              SPOG
 2.  SPOG  (Every position is constrained)                   SPOG
 3.  SPO?  (S, P, and O are constrained; G is not)           SPOG
 4.  SP??  (S and P are constrained; O and G are not)        SPOG
 5.  S???  (S is constrained; P, O, and G are not)           SPOG
 6.  S??G  (S and G are constrained; P and O are not)        SPOG
```

```
 7.  ?POG  (P, O, and G are constrained; S is not)          POGS
 8.  ?PO?  (P and O are constrained; S and G are not)       POGS
 9.  ?P??  (P is constrained; S, O, and G are not)          POGS

10.  ?P?G  (P and G are constrained; S and O are not)       GPSO
11.  SP?G  (S, P, and G are constrained; O is not)          GPSO
12.  ???G  (G is constrained; S, P, and O are not)          GPSO

13.  S?OG  (S, O, and G are constrained; P is not)          OGSP
14.  ??OG  (O and G are constrained; S and P are not)       OGSP
15.  ??O?  (O is constrained; S, P, and G are not)          OGSP

16.  S?O?  (S and O are constrained; P and G are not)       OSGP
```

Neptune menciptakan dan mempertahankan hanya tiga dari keenam indeks tersebut secara default:

- `SPOG` – Menggunakan kunci yang terdiri dari `Subject + Predicate + Object + Graph`.
- `POGS` – Menggunakan kunci yang terdiri dari `Predicate + Object + Graph + Subject`.
- `GPSO` – Menggunakan kunci yang terdiri dari `Graph + Predicate + Subject + Object`.

Ketiga indeks ini menangani banyak pola akses yang paling umum. Mempertahankan hanya tiga indeks pernyataan penuh alih-alih keenamnya sangat mengurangi sumber daya yang Anda butuhkan untuk mendukung akses cepat tanpa pemindaian dan penyaringan. Misalnya, indeks `SPOG` memungkinkan pencarian efisien setiap kali prefiks dari posisi, seperti vertex atau vertex dan pengidentifikasi properti, terikat. Indeks `POGS` memungkinkan akses yang efisien ketika hanya edge atau label properti disimpan dalam posisi `P` terikat.

API tingkat rendah untuk menemukan pernyataan mengambil pola pernyataan di mana beberapa posisi diketahui dan sisanya dibiarkan untuk penemuan oleh pencarian indeks. Dengan menyusun posisi yang dikenal menjadi prefiks kunci sesuai dengan urutan kunci indeks untuk salah satu indeks pernyataan, Neptune melakukan pemindaian rentang untuk mengambil semua pernyataan yang cocok dengan posisi yang diketahui.

Namun, salah satu indeks pernyataan yang tidak dibuat Neptune secara default adalah indeks `OSGP` traversal terbalik, yang dapat mengumpulkan predikat di seluruh objek dan subjek. Sebaliknya, Neptune secara default melacak predikat yang berbeda dalam indeks terpisah yang digunakan untuk melakukan pemindaian gabungan `{all P x POGS}`. Ketika Anda bekerja dengan Gremlin, predikat sesuai dengan properti atau label edge.

Jika jumlah predikat yang berbeda dalam grafik menjadi besar, strategi akses Neptune default dapat menjadi tidak efisien. Di Gremlin, misalnya, sebuah langkah `in()` di mana tidak ada label edge diberikan, atau langkah apa pun yang menggunakan `in()` secara internal seperti `both()` atau `drop()`, bisa menjadi sangat tidak efisien.

Mengaktifkan Pembuatan Indeks OSGP Menggunakan Mode Lab

Jika model data Anda membuat sejumlah besar predikat berbeda, Anda mungkin mengalami penurunan kinerja dan biaya operasional yang lebih tinggi yang dapat ditingkatkan secara dramatis dengan menggunakan Mode Lab untuk mengaktifkan indeks OSGP selain tiga indeks yang dipertahankan Neptunus secara default.

Mengaktifkan indeks OSGP dapat memiliki beberapa sisi negatif:

- Tingkat insert dapat memperlambat hingga 23%.
- Penyimpanan meningkat hingga 20%.
- Kueri baca yang menyentuh semua indeks secara seimbang (yang cukup langka) mungkin memiliki peningkatan latensi.

Namun, secara umum mengaktifkan indeks OSGP untuk Klaster DB dengan jumlah besar predikat yang berbeda sangat menguntungkan. Pencarian berbasis objek menjadi sangat efisien (misalnya, menemukan semua edge yang masuk ke vertex, atau semua subjek yang terhubung ke objek tertentu), dan sebagai hasilnya menjatuhkan vertex juga menjadi jauh lebih efisien.

> **Important**
>
> Anda hanya dapat mengaktifkan indeks OSGP dalam klaster DB kosong, sebelum Anda memuat data ke dalamnya.

Pernyataan Gremlin dalam model data Neptune

Data properti-grafik Gremlin dinyatakan dalam model SPOG menggunakan tiga kelas pernyataan, yaitu:

- Pernyataan Label Vertex Gremlin

- Pernyataan Edge
- Pernyataan Properti

Untuk penjelasan tentang bagaimana ketiganya digunakan dalam kueri Gremlin, lihat Memahami bagaimana kueri Gremlin bekerja di Neptune.

Cache pencarian Neptune dapat mempercepat kueri baca

Amazon Neptune mengimplementasikan cache pencarian yang menggunakan R5d SSD berbasis instans NVMe untuk meningkatkan kinerja baca kueri dengan pencarian nilai properti atau literal RDF yang sering dan berulang. Cache pencarian untuk sementara menyimpan nilai-nilai ini dalam volume NVMe SSD di mana mereka dapat diakses dengan cepat.

Kueri baca yang mengembalikan properti sejumlah besar vertex dan edge, atau banyak tripel RDF, dapat memiliki latensi tinggi jika nilai properti atau literalnya perlu diambil dari volume penyimpanan klaster alih-alih memori. Contohnya termasuk kueri baca berjalan lama yang mengembalikan sejumlah besar nama lengkap dari grafik identitas, atau alamat IP dari grafik deteksi penipuan. Saat jumlah nilai properti atau literal RDF yang dikembalikan oleh kueri Anda meningkat, memori yang tersedia berkurang dan eksekusi kueri Anda dapat secara signifikan menurun.

Kasus penggunaan untuk cache pencarian Neptunus

Cache pencarian hanya membantu ketika kueri baca Anda mengembalikan properti dari sejumlah besar simpul dan tepi, atau tiga kali lipat RDF.

Untuk mengoptimalkan kinerja kueri, Amazon Neptunus menggunakan R5d jenis instans untuk membuat cache besar untuk nilai properti atau literal tersebut. Mengambilnya dari cache kemudian jauh lebih cepat daripada mengambilnya dari volume penyimpanan cluster.

Sebagai aturan praktis, hanya bermanfaat untuk mengaktifkan cache pencarian jika ketiga kondisi berikut terpenuhi:

- Anda telah mengamati peningkatan latensi dalam kueri baca.
- Anda juga mengamati penurunan `BufferCacheHitRatio` CloudWatch metrik saat menjalankan kueri baca (lihatMemantau Neptunus Menggunakan Amazon CloudWatch).
- Kueri baca Anda menghabiskan banyak waktu dalam mematerialisasi nilai kembali sebelum me-render hasilnya (lihat contoh profil Gremlin di bawah ini untuk cara untuk menentukan berapa banyak nilai properti yang sedang dimaterialisasi untuk kueri).

> **Note**
>
> Fitur ini hanya membantu dalam skenario spesifik yang dijelaskan di atas. Misalnya, cache pencarian tidak membantu kueri agregasi sama sekali. Kecuali Anda menjalankan kueri yang

akan mendapat manfaat dari cache pencarian, tidak ada alasan untuk menggunakan tipe R5d instance alih-alih jenis instance yang setara dan lebih murahR5.

Jika Anda menggunakan Gremlin, Anda dapat menilai biaya materialisasi kueri dengan API `profile` Gremlin. Di bawah “Operasi Indeks', ia menunjukkan jumlah istilah yang termaterialisasi selama eksekusi:

```
Index Operations
Query execution:
    # of statement index ops: 3
    # of unique statement index ops: 3
    Duplication ratio: 1.0
    # of terms materialized: 5273
Serialization:
    # of statement index ops: 200
    # of unique statement index ops: 140
    Duplication ratio: 1.43
    # of terms materialized: 32693
```

Jumlah istilah non-numerik yang termaterialisasi berbanding lurus dengan jumlah istilah pencarian yang harus dilakukan Neptune.

Menggunakan cache pencarian

Cache pencarian hanya tersedia pada jenis R5d instance, di mana cache ini diaktifkan secara otomatis secara default. Instans R5d Neptunus memiliki spesifikasi R5 yang sama dengan instans, ditambah hingga 1,8 TB penyimpanan SSD berbasis lokal. NVMe Cache pencarian adalah khusus instans, dan beban kerja yang mendapat keuntungan dapat diarahkan secara khusus ke instans R5d dalam sebuah klaster Neptune, sementara beban kerja lainnya dapat diarahkan ke R5 atau tipe instans lainnya.

Untuk menggunakan cache pencarian pada instance Neptunus, cukup tingkatkan instance itu ke jenis instance. R5d Ketika Anda melakukannya, Neptunus secara otomatis menyetel parameter cluster DB 1 ke (diaktifkan), dan membuat cache pencarian pada instance tertentu. neptune_lookup_cache Anda kemudian dapat menggunakan Status instans API untuk mengonfirmasi bahwa cache telah diaktifkan.

Demikian pula, untuk menonaktifkan cache pencarian pada instance tertentu, turunkan instance dari tipe R5d instance ke tipe R5 instance yang setara.

Saat instans R5d diluncurkan, cache pencarian diaktifkan dan dalam mode start dingin, yang berarti bahwa cache kosong. Neptune pertama memeriksa di dalam cache pencarian untuk nilai properti atau literal RDF saat memproses kueri, dan menambahkan nilai dan literal jika belum ada. Ini secara beransur-ansur memanaskan cache.

Ketika Anda mengarahkan kueri baca yang memerlukan pencarian nilai properti atau literal RDF untuk instans pembaca R5d, performa baca menurun sedikit sementara cache memanas. Namun, ketika cache dihangatkan, kinerja baca meningkat secara signifikan dan Anda mungkin juga melihat penurunan I/O biaya yang terkait dengan pencarian yang mengenai cache daripada penyimpanan cluster. Pemanfaatan memori juga meningkat.

Jika instans penulis adalah sebuah R5d, instans tersebut memanaskan cache pencariannya secara otomatis pada setiap operasi tulis. Pendekatan ini sedikit meningkatkan latensi untuk kueri tulis, tetapi memanaskan cache pencarian dengan lebih efisien. Kemudian jika Anda mengarahkan kueri baca yang memerlukan pencarian nilai properti atau literal RDF ke instans penulis, Anda mulai mendapatkan peningkatan performa baca dengan segera, karena nilai-nilai telah di-cache di sana.

Selain itu, jika Anda menjalankan loader massal pada instans tulis R5d, Anda mungkin memperhatikan bahwa performanya sedikit terdegradasi karena cache.

Karena cache pencarian dikhususkan untuk setiap node, penggantian host me-reset cache ke start dingin.

Anda dapat menonaktifkan sementara cache pencarian pada semua instance di cluster DB Anda dengan menyetel parameter cluster neptune_lookup_cache DB ke 0 (dinonaktifkan). Secara umum, bagaimanapun, lebih masuk akal untuk menonaktifkan cache pada instance tertentu dengan menurunkannya dari tipe R5 instance R5d ke bawah.

Semantik Transaksi di Neptune

Amazon Neptune dirancang untuk mendukung beban kerja pemrosesan transaksional online yang bersamaan (OLTP) melalui grafik data. W3C SPARQL Query Language untuk spesifikasi RDF dan dokumentasi Apache TinkerPop Gremlin Graph Traversal Language tidak mendefinisikan semantik transaksi untuk pemrosesan kueri bersamaan. Karena dukungan ACID dan jaminan transaksi yang terdefinisi dengan baik bisa sangat penting, kami menerapkan semantik yang ketat untuk membantu menghindari anomali data.

Bagian ini mendefinisikan semantik ini dan menggambarkan bagaimana mereka berlaku untuk beberapa kasus penggunaan umum di Neptunus.

Topik

- Definisi Tingkat Isolasi
- Tingkat Isolasi Transaksi di Neptune
- Contoh semantik transaksi Neptunus

Definisi Tingkat Isolasi

“I” di `ACID` singkatan dari Isolasi. Tingkat isolasi transaksi menentukan berapa banyak atau sedikit transaksi bersamaan lainnya dapat mempengaruhi data yang beroperasi.

SQL:1992 standar membuat perbendaharaan kata untuk menerangkan tingkat isolasi. Ini mendefinisikan tiga jenis interaksi (yang disebut fenomena) yang dapat terjadi antara dua transaksi bersamaan, `Tx1` dan `Tx2`:

- `Dirty read` – Hal ini terjadi ketika `Tx1` memodifikasi item, dan kemudian `Tx2` membaca item tersebut sebelum `Tx1` telah melakukan perubahan. Kemudian, jika `Tx1` tidak pernah berhasil melakukan perubahan, atau kembali, `Tx2` telah membaca nilai yang tidak pernah berhasil masuk ke dalam database.
- `Non-repeatable read` – Hal ini terjadi ketika `Tx1` membaca item, maka `Tx2` memodifikasi atau menghapus item tersebut dan melakukan perubahan, kemudian `Tx1` mencoba membaca ulang item tersebut. `Tx1` sekarang membaca nilai yang berbeda dari sebelumnya, atau menemukan bahwa item tidak lagi ada.
- `Phantom read` – Hal ini terjadi ketika `Tx1` membaca satu set item yang memenuhi kriteria pencarian, dan kemudian `Tx2` menambahkan item baru yang memenuhi kriteria pencarian,

kemudian `Tx1` mengulangi pencarian. `Tx1` sekarang memperoleh satu set item yang berbeda daripada sebelumnya.

Masing-masing dari tiga jenis interaksi dapat menyebabkan inkonsistensi dalam data yang dihasilkan dalam database.

SQL:1992 standar mendefinisikan empat tingkat isolasi yang memiliki jaminan yang berbeda dalam hal tiga jenis interaksi dan inkonsistensi yang dapat mereka hasilkan. Pada keempat tingkat, transaksi dapat dijamin untuk mengeksekusi sepenuhnya atau tidak sama sekali:

- `READ UNCOMMITTED` – Memungkinkan semua tiga jenis interaksi (yaitu, dirty reads, non-repeatable reads, dan phantom reads).
- `READ COMMITTED` – Dirty reads sifatnya tidak mungkin, tapi nonrepeatable dan phantom reads mungkin.
- `REPEATABLE READ` – Tidak satupun dari dirty read dan nonrepeatable read sifatnya mungkin, tetapi phantom read masih mungkin.
- `SERIALIZABLE` – Tak satu pun dari tiga jenis fenomena interaksi dapat terjadi.

Multiversion concurrency control (MVCC) memungkinkan satu jenis lain dari isolasi, yaitu SNAPSHOT isolasi. Ini menjamin bahwa transaksi beroperasi pada snapshot data seperti saat transaksi dimulai, dan bahwa tidak ada transaksi lain dapat mengubah snapshot itu.

Tingkat Isolasi Transaksi di Neptune

Amazon Neptune mengimplementasikan tingkat isolasi transaksi yang berbeda untuk kueri baca-saja dan kueri mutasi. Kueri SPARQL dan Gremlin diklasifikasikan sebagai baca-saja atau mutasi berdasarkan kriteria sebagai berikut:

- Di SPARQL, ada perbedaan yang jelas antara kueri baca (`SELECT`, `ASK`, `CONSTRUCT`, dan `DESCRIBE` sebagaimana didefinisikan dalam spesifikasi Bahasa Kueri SPARQL 1.1), dan kueri mutasi (`INSERT` dan `DELETE` sebagaimana didefinisikan dalam spesifikasi Pembaruan SPARQL 1.1).

  Perhatikan bahwa Neptune memperlakukan beberapa kueri mutasi dikirimkan bersama-sama (misalnya, dalam pesan `POST`, dipisahkan dengan titik koma) sebagai transaksi tunggal. Mereka dijamin berhasil atau gagal sebagai unit atom, dan dalam kasus kegagalan, perubahan parsial digulung kembali.

- Namun, di Gremlin, Neptunus mengklasifikasikan kueri sebagai kueri hanya-baca atau kueri mutasi berdasarkan apakah kueri berisi langkah-langkah jalur kueri seperti,,, atau yang memanipulasi data. `addE()` `addV()` `property()` `drop()` Jika kueri berisi langkah jalur apa pun, maka diklasifikasikan dan dieksekusi sebagai kuery mutasi.

Hal ini juga memungkinkan untuk menggunakan sesi berdiri di Gremlin. Untuk informasi selengkapnya, lihat Sesi berbasis skrip Gremlin. Dalam sesi ini, semua kueri, termasuk kueri hanya-baca, dijalankan di bawah isolasi yang sama dengan kueri mutasi pada titik akhir penulis.

Menggunakan sesi baca-tulis baut di OpenCypher, semua kueri termasuk kueri hanya-baca dijalankan di bawah isolasi yang sama dengan kueri mutasi, pada titik akhir penulis.

Topik

- Isolasi kueri hanya-baca di Neptunus
- Isolasi kueri mutasi di Neptunus
- Resolusi Konflik Menggunakan Lock-Wait Timeout
- Kunci rentang dan konflik palsu

Isolasi kueri hanya-baca di Neptunus

Neptune mengevaluasi kueri baca-saja di bawah semantik isolasi snapshot. Ini berarti bahwa kueri baca saja secara logis beroperasi pada snapshot yang konsisten dari database yang diambil ketika evaluasi kueri dimulai. Neptune kemudian dapat menjamin bahwa tidak ada fenomena berikut yang akan terjadi:

- `Dirty reads` – Kueri baca-saja di Neptune tidak akan pernah melihat data yang tidak terikat dari transaksi bersamaan.
- `Non-repeatable reads` – Sebuah transaksi baca-saja yang membaca data yang sama lebih dari sekali akan selalu mendapatkan nilai yang sama.
- `Phantom reads` – Sebuah transaksi baca-saja tidak akan pernah membaca data yang ditambahkan setelah transaksi dimulai.

Karena isolasi snapshot dicapai menggunakan multiversion concurrency control (MVCC), kueri baca-saja tidak perlu mengunci data dan karena itu tidak memblokir kueri mutasi.

Replika baca hanya menerima kueri baca-saja, sehingga semua kueri yang berlawanan dengan replika baca dieksekusi replika baca di bawah semantik isolasi SNAPSHOT.

Satu-satunya pertimbangan tambahan ketika meng-kueri replika baca adalah bahwa ada lag replikasi kecil antara penulis dan replika baca. Ini berarti bahwa pembaruan yang dibuat pada penulis mungkin mengambil waktu singkat untuk disebarkan ke replika baca yang sedang Anda baca. Waktu replikasi sebenarnya tergantung pada pemuatan tulis terhadap instance utama. Arsitektur Neptunus mendukung replikasi latensi rendah dan lag replikasi diinstrumentasi dalam metrik Amazon. CloudWatch

Namun, karena tingkat isolasi SNAPSHOT, kueri baca selalu melihat keadaan yang konsisten dari database, bahkan jika itu bukan yang terbaru.

Dalam kasus ketika Anda memerlukan jaminan yang kuat bahwa kueri mengamati hasil dari pembaruan sebelumnya, kirim kueri ke titik akhir penulis itu sendiri alih-alih ke replika baca.

Isolasi kueri mutasi di Neptunus

Pembacaan yang dibuat sebagai bagian dari kueri mutasi dijalankan di bawah isolasi transaksi `READ COMMITTED`, yang mengeliminasi kemungkinan dirty-reads. Melampaui jaminan biasa yang disediakan untuk isolasi transaksi `READ COMMITTED`, Neptune memberikan jaminan yang kuat bahwa baik pembacaan `NON-REPEATABLE` dan `PHANTOM` bisa terjadi.

Jaminan kuat ini dicapai dengan mengunci catatan dan rentang catatan saat membaca data. Hal ini mencegah transaksi bersamaan dari membuat penyisipan atau penghapusan dalam rentang indeks setelah mereka baca, sehingga menjamin pembacaan berulang.

> **Note**
>
> Namun, transaksi mutasi bersamaan `Tx2` bisa dimulai setelah dimulainya transaksi mutasi `Tx1`, dan dapat melakukan perubahan sebelum `Tx1` mengunci data untuk dibaca. Dalam kasus itu, `Tx1` akan melihat perubahan `Tx2` seperti jika `Tx2` telah selesai sebelum `Tx1`dimulai. Karena ini hanya berlaku untuk perubahan yang dilakukan, `dirty read` tidak akan pernah terjadi.

Untuk memahami mekanisme penguncian yang digunakan Neptune untuk kueri mutasi, terlebih dahulu itu membantu memahami detail Neptune Model Data Grafik dan Strategi Pengindeksan. Neptune mengelola data menggunakan tiga indeks, yaitu SPOG, POGS, dan GPSO.

Untuk mencapai pembacaan berulang untuk tingkat transaksi `READ COMMITTED`, Neptune mengambil kunci rentang dalam indeks yang sedang digunakan. Sebagai contoh, jika kueri mutasi membaca semua properti dan edge keluar dari sebuah vertex bernama `person1`, simpul akan mengunci seluruh rentang yang didefinisikan oleh awalan `S=person1` dalam indeks `SPOG` sebelum membaca data.

Mekanisme yang sama berlaku saat menggunakan indeks lain. Sebagai contoh, ketika transaksi mutasi mencari semua pasangan vertex sumber-target untuk label edge yang diberikan menggunakan indeks `POGS`, rentang label edge dalam posisi `P` akan terkunci. Setiap transaksi bersamaan, terlepas dari apakah itu baca-saja atau kueri mutasi, masih bisa melakukan pembacaan dalam rentang yang terkunci. Namun, setiap mutasi yang melibatkan penyisipan atau penghapusan catatan baru dalam rentang prefiks terkunci akan memerlukan kunci eksklusif dan akan dicegah.

Dengan kata lain, ketika rentang indeks telah dibaca oleh transaksi mutasi, ada jaminan kuat bahwa rentang ini tidak akan dimodifikasi oleh transaksi bersamaan apapun sampai akhir transaksi pembacaan. Ini menjamin bahwa tidak ada `non-repeatable reads` akan terjadi.

Resolusi Konflik Menggunakan Lock-Wait Timeout

Jika transaksi kedua mencoba memodifikasi catatan dalam rentang tempat transaksi pertama telah terkunci, Neptune mendeteksi konflik dengan segera dan memblokir transaksi kedua.

Jika tidak ada deadlock dependensi terdeteksi, Neptune secara otomatis memberlakukan mekanisme timeout lock-wait, di mana transaksi yang diblokir menunggu transaksi yang menahan kunci hingga 60 detik untuk diselesaikan dan dilepaskan kuncinya.

- Jika timeout lock-wait berakhir sebelum kunci dirilis, transaksi yang diblokir akan diroll-back.
- Jika kunci dilepaskan dalam timeout lock-wait, transaksi kedua tidak diblokir dan berhasil menyelesaikan tanpa perlu mencoba lagi.

Namun, jika Neptune mendeteksi deadlock dependensi diantara dua transaksi, rekonsiliasi otomatis konflik tidak dimungkinkan. Dalam hal ini, Neptunus segera membatalkan dan memutar kembali salah satu dari dua transaksi tanpa memulai batas waktu tunggu kunci-tunggu. Neptunus melakukan upaya terbaik untuk memutar kembali transaksi yang memiliki catatan paling sedikit dimasukkan atau dihapus.

Mengukur waktu kunci-tunggu (mesin ≥ 1.4.5.0)

Dimulai dengan engine versi 1.4.5.0, Anda dapat mengamati dengan tepat berapa lama kueri mutasi diblokir dengan menggunakan dua penghitung: slow-query-log

Penghitung	Deskripsi
`sharedLocksWaitTimeMillis`	Waktu yang dihabiskan menunggu untuk mendapatkan kunci bersama (S), yang memungkinkan banyak pembaca tetapi memblokir penulis.
`exclusiveLocksWaitTimeMillis`	Waktu yang dihabiskan menunggu untuk mendapatkan kunci eksklusif (X), yang memblokir semua akses lainnya.

Kedua bidang ini muncul di `storageCounters` objek hanya ketika Anda mengaktifkan logging permintaan lambat dalam `debug` mode ()`neptune_enable_slow_query_log=debug`.

> **Tip**
>
> Jika `sharedLocksWaitTimeMillis + exclusiveLocksWaitTimeMillis` mendekati kuerio`verallRunTimeMs`, kueri terhambat oleh pertentangan kunci daripada CPU, jaringan, atau I/O.

Kiat praktis untuk mengurangi pertengkaran:

- Pekerjaan yang saling bertentangan — Jalankan mutasi batch berat selama periode aktivitas pengguna yang lebih rendah.
- Pecahkan mutasi besar menjadi potongan yang lebih kecil — Transaksi yang lebih kecil menahan kunci untuk waktu yang lebih singkat, mengurangi kemungkinan batas waktu.

Kunci rentang dan konflik palsu

Neptunus mengambil kunci jangkauan menggunakan kunci celah. Kunci celah adalah kunci pada celah antara catatan indeks, atau kunci pada celah sebelum catatan indeks pertama atau setelah catatan indeks terakhir.

Neptunus menggunakan tabel kamus yang disebut untuk mengaitkan nilai ID numerik dengan literal string tertentu. Berikut adalah contoh keadaan kamus Neptunus seperti itu: tabel:

String	ID
jenis	1
default_graph	2
orang_3	3
orang_1	5
tahu	6
orang_2	7
usia	8
tepi_1	9
lives_in	10
New York	11
Orang	12
Tempat	13
tepi_2	14

String di atas termasuk dalam model grafik properti, tetapi konsep berlaku sama untuk semua model grafik RDF juga.

Keadaan yang sesuai dari indeks SPOG (Subject-Predicate-Object_Graph) ditunjukkan di bawah di sebelah kiri. Di sebelah kanan, string yang sesuai ditampilkan, untuk membantu memahami apa arti data indeks.

S (ID)	P (ID)	O (ID)	G (ID)		S (string)	P (string)	O (string)	G (string)
3	1	12	2		orang_3	jenis	Orang	default_graph
5	1	12	2		orang_1	jenis	Orang	default_graph
5	6	3	9		orang_1	tahu	orang_3	tepi_1
5	8	40	2		orang_1	usia	40	default_graph
5	10	11	14		orang_1	lives_in	New York	tepi_2
7	1	12	2		orang_2	jenis	Orang	default_graph
11	1	13	2		New York	jenis	Tempat	default_graph

Sekarang, jika kueri mutasi membaca semua properti dan tepi keluar dari simpul bernama`person_1`, node akan mengunci seluruh rentang yang ditentukan oleh awalan `S=person_1` dalam indeks SPOG sebelum membaca data. Kunci rentang akan menempatkan kunci celah pada semua catatan yang cocok dan catatan pertama yang tidak cocok. Catatan yang cocok akan dikunci, dan catatan yang tidak cocok tidak akan dikunci. Neptunus akan menempatkan kunci celah sebagai berikut:

- `5 1 12 2` (celah 1)
- `5 6 3 9` (celah 2)
- `5 8 40 2` (celah 3)
- `5 10 11 14` (celah 4)
- `7 1 12 2` (celah 5)

Ini mengunci catatan berikut:

- `5 1 12 2`
- `5 6 3 9`
- `5 8 40 2`
- `5 10 11 14`

Dalam keadaan ini, operasi berikut diblokir secara sah:

- Penyisipan properti atau tepi baru untuk`S=person_1`. Properti baru yang berbeda dari `type` atau tepi baru harus masuk ke celah 2, celah 3, celah 4, atau celah 5, yang semuanya terkunci.
- Penghapusan salah satu catatan yang ada.

Pada saat yang sama, beberapa operasi bersamaan akan diblokir secara salah (menghasilkan konflik palsu):

- Setiap properti atau penyisipan tepi untuk `S=person_3` diblokir karena harus masuk ke celah 1.
- Setiap penyisipan simpul baru yang diberi ID antara 3 dan 5 akan diblokir karena harus masuk ke celah 1.
- Setiap penyisipan simpul baru yang diberi ID antara 5 dan 7 akan diblokir karena harus masuk ke celah 5.

Kunci celah tidak cukup tepat untuk mengunci celah untuk satu predikat tertentu (misalnya, untuk mengunci gap5 untuk predikat). `S=5`

Kunci rentang hanya ditempatkan di indeks tempat pembacaan terjadi. Dalam kasus di atas, catatan dikunci hanya dalam indeks SPOG, bukan di POGS atau GPSO. Pembacaan untuk kueri dapat dilakukan di semua indeks tergantung pada pola akses, yang dapat dicantumkan menggunakan `explain` APIs (untuk Sparql dan untuk Gremlin).

> **Note**
>
> Kunci celah juga dapat diambil untuk pembaruan bersamaan yang aman pada indeks yang mendasarinya, yang juga dapat menyebabkan konflik palsu. Kunci celah ini ditempatkan independen dari tingkat isolasi atau operasi baca yang dilakukan oleh transaksi.

Konflik palsu dapat terjadi tidak hanya ketika transaksi bersamaan bertabrakan karena kunci celah, tetapi juga dalam beberapa kasus ketika transaksi sedang dicoba lagi setelah segala jenis kegagalan. Jika roll-back yang dipicu oleh kegagalan masih berlangsung dan kunci yang sebelumnya diambil untuk transaksi belum sepenuhnya dirilis, percobaan ulang akan menghadapi konflik palsu dan gagal.

Di bawah beban tinggi, Anda mungkin biasanya menemukan bahwa 3-4% kueri tulis gagal karena konflik palsu. Untuk klien eksternal, konflik palsu semacam itu sulit diprediksi, dan harus ditangani menggunakan percobaan ulang.

Contoh semantik transaksi Neptunus

Contoh berikut menggambarkan kasus penggunaan yang berbeda untuk semantik transaksi di Amazon Neptune.

Topik

- Contoh 1 - Memasukkan Properti Hanya Jika Tidak Ada
- Contoh 2 - Menegaskan Bahwa Nilai Properti Adalah Unik secara Global
- Contoh 3 - Mengubah Properti Jika Properti Lain Memiliki Nilai Tertentu
- Contoh 4 - Mengganti Properti yang Ada
- Contoh 5 - Menghindari Properti atau Edge Menggantung

Contoh 1 - Memasukkan Properti Hanya Jika Tidak Ada

Misalkan Anda ingin memastikan bahwa properti diatur hanya sekali. Sebagai contoh, misalkan beberapa kueri mencoba untuk menetapkan kepada seseorang skor kredit secara bersamaan. Anda hanya ingin satu instans dari properti dimasukkan, dan kueri lainnya gagal karena properti telah ditetapkan.

```
# GREMLIN:
g.V('person1').hasLabel('Person').coalesce(has('creditScore'), property('creditScore',
 'AAA+'))

# SPARQL:
INSERT { :person1 :creditScore "AAA+" .}
WHERE  { :person1 rdf:type :Person .
         FILTER NOT EXISTS { :person1 :creditScore ?o .} }
```

Langkah `property()` Gremlin menyisipkan properti dengan kunci dan nilai yang diberikan. Langkah `coalesce()` mengeksekusi argumen pertama di langkah pertama, dan jika gagal, maka mengeksekusi langkah kedua:

Sebelum memasukkan nilai properti `creditScore` untuk vertex `person1` yang diberikan, transaksi harus coba membaca kemungkinan tidak adanya nilai `creditScore` untuk `person1`. Ini mencoba membaca kunci rentang SP untuk `S=person1` dan `P=creditScore` dalam SPOG indeks tempat nilai `creditScore` ada atau akan ditulis.

Mengambil kunci rentang ini mencegah transaksi bersamaan apapun dari memasukkan nilai `creditScore` secara bersamaan. Ketika ada beberapa transaksi paralel, paling banyak salah satu dari transaksi tersebut dapat memperbarui nilai pada suatu waktu. Ini mengeliminasi anomali lebih dari satu properti `creditScore` yang dibuat.

Contoh 2 - Menegaskan Bahwa Nilai Properti Adalah Unik secara Global

Misalkan Anda ingin memasukkan seseorang dengan nomor Jaminan Sosial sebagai kunci utama. Anda ingin kueri mutasi Anda menjamin bahwa, pada tingkat global, tidak ada orang lain dalam database memiliki nomor Jaminan Sosial yang sama:

```
# GREMLIN:
g.V().has('ssn', 123456789).fold()
  .coalesce(__.unfold(),
            __.addV('Person').property('name', 'John Doe').property('ssn', 123456789'))

# SPARQL:
INSERT { :person1 rdf:type :Person .
         :person1 :name "John Doe" .
         :person1 :ssn 123456789 .}
WHERE  { FILTER NOT EXISTS { ?person :ssn 123456789 } }
```

Contoh ini sama dengan yang sebelumnya. Perbedaan utamanya adalah bahwa kunci rentang diambil pada indeks POGS ketimbang indeks SPOG.

Transaksi mengeksekusi kueri harus membaca pola, `?person :ssn 123456789`, di mana posisi P dan 0 terikat. Kunci rentang diambil pada indeks POGS untuk `P=ssn` dan `0=123456789`.

- Jika pola itu ada, tidak ada tindakan yang diambil.
- Jika tidak ada, penguncian mencegah setiap transaksi bersamaan dari menyisipkan nomor Jaminan Sosial juga

Contoh 3 - Mengubah Properti Jika Properti Lain Memiliki Nilai Tertentu

Misalkan berbagai peristiwa dalam game memindahkan seseorang dari tingkat satu ke tingkat dua, dan menugaskan mereka properti `level2Score` baru yang diatur ke nol. Anda harus yakin bahwa beberapa instans yang bersamaan dari transaksi tersebut tidak dapat membuat beberapa instans dari properti skor tingkat-dua. Kueri di Gremlin dan SPARQL mungkin terlihat seperti berikut ini.

```
# GREMLIN:
g.V('person1').hasLabel('Person').has('level', 1)
 .property('level2Score', 0)
 .property(Cardinality.single, 'level', 2)

# SPARQL:
DELETE { :person1 :level 1 .}
INSERT { :person1 :level2Score 0 .
         :person1 :level 2 .}
WHERE  { :person1 rdf:type :Person .
         :person1 :level 1 .}
```

Di Gremlin, ketika `Cardinality.single` ditentukan, langkah `property()` menambahkan properti baru atau menggantikan nilai properti yang ada dengan nilai baru yang ditentukan.

Setiap pembaruan ke nilai properti, seperti meningkatkan `level` dari 1 sampai 2, diimplementasikan sebagai penghapusan catatan saat ini dan penyisipan catatan baru dengan nilai properti baru. Dalam hal ini, catatan dengan tingkat nomor 1 dihapus dan catatan dengan tingkat nomor 2 dimasukkan kembali.

Untuk transaksi agar dapat menambah `level2Score` dan memperbarui `level` dari 1 sampai 2, transaksi harus terlebih dahulu memvalidasi bahwa nilai `level` saat ini sama dengan 1. Dengan demikian, dibutuhkan kunci rentang pada prefiks SPO untuk `S=person1`, `P=level`, dan `O=1` dalam indeks SPOG. Kunci ini mencegah transaksi yang bersamaan menghapus versi 1 tiga kali lipat, dan sebagai hasilnya, tidak ada pembaruan bersamaan yang bertentangan dapat terjadi.

Contoh 4 - Mengganti Properti yang Ada

Peristiwa tertentu dapat memperbarui nilai kredit seseorang ke nilai baru (di sini BBB). Tetapi Anda ingin memastikan bahwa peristiwa bersamaan dari jenis tersebut tidak dapat membuat beberapa properti skor kredit untuk seseorang.

```
# GREMLIN:
```

```
g.V('person1').hasLabel('Person')
 .sideEffect(properties('creditScore').drop())
 .property('creditScore', 'BBB')

# SPARQL:
DELETE { :person1 :creditScore ?o .}
INSERT { :person1 :creditScore "BBB" .}
WHERE  { :person1 rdf:type :Person .
         :person1 :creditScore ?o .}
```

Kasus ini mirip dengan contoh 3, kecuali bahwa alih-alih mengunci SPO awalan, Neptunus mengunci SP awalan dengan dan hanya. S=person1 P=creditScore Hal ini mencegah transaksi bersamaan dari memasukkan atau menghapus setiap triple dengan properti `creditScore` untuk subjek `person1`.

Contoh 5 - Menghindari Properti atau Edge Menggantung

Pembaruan pada entitas tidak harus meninggalkan elemen menggantung, yaitu properti atau edge yang terkait dengan entitas yang tidak diketik. Ini hanya masalah di SPARQL, karena Gremlin memiliki kendala built-in untuk mencegah elemen menggantung .

```
# SPARQL:
tx1: INSERT { :person1 :age 23 } WHERE { :person1 rdf:type :Person }
tx2: DELETE { :person1 ?p ?o }
```

Kueri `INSERT` harus membaca dan mengunci prefiks SPO dengan S=person1, P=rdf:type, dan O=Person dalam indeks SPOG. Penguncian mencegah kueri `DELETE` dari berhasilnya secara paralel.

Dalam perlombaan antara query `DELETE` mencoba untuk menghapus rekaman `:person1 rdf:type :Person` dan query `INSERT` membaca catatan dan membuat kunci rentang di SPO dalam Indeks SPOG, hasil berikut ini mungkin:

- Jika kueri `INSERT` melakukan sebelum kueri `DELETE` membaca dan menghapus semua catatan untuk `:person1`, `:person1` dihapus seluruhnya dari database, termasuk catatan yang baru dimasukkan.
- Jika query `DELETE` melakukan sebelum kueri `INSERT` coba membaca catatan `:person1 rdf:type :Person`, pembacaan mengamati perubahan yang dilakukan. Artinya, tidak menemukan catatan `:person1 rdf:type :Person` apapun dan karenanya menjadi no-op.

- Jika query `INSERT` membaca sebelum query `DELETE`, triple `:person1 rdf:type :Person` terkunci dan kueri `DELETE` diblokir sampai kueri INSERT dilakukan, seperti dalam kasus pertama sebelumnya.
- Jika bacaan `DELETE` membaca sebelum kueri `INSERT`, dan kueri `INSERT` mencoba untuk membaca dan mengunci prefiks `SPO` untuk catatan, maka konflik terdeteksi. Hal ini karena triple telah ditandai untuk dihapus, dan `INSERT` kemudian gagal.

Dalam semua urutan kemungkinan yang berbeda dari peristiwa ini, tidak ada edge menggantung dibuat.

Klaster dan Instans DB Amazon Neptune

Klaster DB Amazon Neptune mengelola akses ke data Anda melalui kueri. Sebuah klaster terdiri dari:

- Satu Instans DB primer..
- Hingga 15 instans DB replika baca..

Semua instans dalam sebuah klaster berbagi volume penyimpanan terkelola yang mendasari yang sama, yang didesain untuk keandalan dan ketersediaan yang tinggi.

Anda terhubung ke instans DB di klaster DB Anda melalui titik akhir Neptune.

Instans DB primer dalam klaster DB Neptune

Instans DB primer mengoordinasikan semua operasi tulis ke volume penyimpanan yang mendasari klaster DB. Ini juga mendukung operasi baca.

Hanya boleh ada satu instans DB primer dalam klaster DB Neptune. Jika instans primer menjadi tidak tersedia, Neptune secara otomatis melakukan failover ke salah satu instans replika baca dengan prioritas yang dapat Anda tentukan.

Instans DB replika baca dalam sebuah klaster DB Neptune

Setelah Anda membuat instans primer untuk klaster DB, Anda dapat membuat hingga 15 instans replika baca dalam klaster DB.

Instans DB replika baca Neptune berfungsi dengan baik untuk penyekalaan kapasitas baca karena didedikasikan sepenuhnya untuk operasi baca pada volume klaster Anda. Semua operasi tulis dikelola oleh instans primer. Setiap instans DB replika baca memiliki titik akhir sendiri.

Karena volume penyimpanan klaster dibagi di antara semua contoh dalam sebuah klaster, semua instans replika data mengembalikan data yang sama untuk hasil kueri dengan sangat sedikit keterlambatan replikasi. Keterlambatan ini biasanya kurang dari 100 milidetik setelah instans primer menulis pembaruan, meskipun dapat sedikit lebih lama ketika volume operasi tulis sangat besar.

Memiliki satu atau beberapa instans replika baca yang tersedia di Availability Zones yang berbeda dapat meningkatkan ketersediaan, karena replika baca berfungsi sebagai target failover untuk instans primer. Artinya, jika instans primer gagal, Neptune mempromosikan instans replika baca menjadi

instans primer. Saat ini terjadi, terdapat gangguan singkat saat instans yang dipromosikan direboot, selama permintaan baca dan tulis dibuat ke instans primer gagal dengan pengecualian.

Sebaliknya, jika klaster DB Anda tidak menyertakan instans replika baca, klaster DB Anda tetap tidak tersedia ketika instans primer gagal sampai telah dibuat ulang. Membuat ulang instans primer membutuhkan waktu lebih lama daripada mempromosikan replika baca.

Untuk memastikan ketersediaan tinggi, kami sarankan Anda membuat satu atau beberapa instans replika baca yang memiliki kelas instans DB yang sama seperti instans primer dan terletak di Availability Zones yang berbeda daripada instans primer. Lihat Toleransi kesalahan untuk cluster DB Neptunus.

Dengan menggunakan konsol , Anda dapat membuat penerapan Multi-AZ cukup dengan menentukan Multi-AZ saat membuat klaster DB. Jika klaster DB berada di satu Availability Zone, Anda dapat menjadikannya klaster DB multi-AZ yang menambahkan replika Neptune di Availability Zone yang berbeda.

> **Note**
>
> Anda tidak dapat membuat instans replika baca terenkripsi untuk klaster Neptune DB yang tidak terenkripsi, atau instans replika baca yang tidak dienkripsi untuk klaster Neptune DB terenkripsi.

Untuk detail tentang cara membuat instans DB replika baca Neptune, lihat Membuat instance pembaca Neptunus menggunakan konsol.

Mengukur instans DB dalam sebuah klaster Neptune DB

Ukur instans di klaster Neptune DB Anda berdasarkan kebutuhan CPU dan memori Anda. Jumlah v CPUs pada sebuah instance menentukan jumlah thread query yang menangani query masuk. Jumlah memori pada instans menentukan ukuran cache buffer, digunakan untuk menyimpan salinan halaman data yang diambil dari volume penyimpanan yang mendasari.

Setiap instans DB Neptunus memiliki sejumlah thread kueri yang sama dengan 2 x jumlah CPUs v pada instance itu. An`r5.4xlarge`, misalnya, dengan 16 vCPUs, memiliki 32 utas kueri, dan karenanya dapat memproses 32 kueri secara bersamaan.

Kueri tambahan yang tiba saat semua utas kueri ditempati akan dimasukkan ke dalam antrean sisi server, dan diproses dengan cara FIFO saat utas kueri menjadi tersedia. Antrean sisi server

ini dapat menyimpan sekitar 8000 permintaan tertunda. Setelah penuh, Neptune menanggapi permintaan tambahan dengan `ThrottlingException`. Anda dapat memantau jumlah permintaan yang tertunda dengan `MainRequestQueuePendingRequests` CloudWatch metrik, atau dengan menggunakan titik akhir status kueri Gremlin dengan parameter. `includeWaiting`

Waktu eksekusi kueri dari perspektif klien menyertakan setiap waktu yang dihabiskan dalam antrean, selain waktu yang dibutuhkan untuk benar-benar mengeksekusi kueri.

Sebuah pemuatan tulis bersamaan yang berkelanjutan yang memanfaatkan semua utas kueri pada instans DB primer idealnya menunjukkan 90% utilisasi CPU atau lebih, yang menunjukkan bahwa semua utas kueri pada server secara aktif terlibat dalam melakukan pekerjaan yang berguna. Namun, utilisasi CPU sebenarnya sering agak rendah, bahkan di bawah pemuatan tulis bersamaan berkelanjutan. Ini biasanya karena utas kueri menunggu I/O operasi hingga volume penyimpanan yang mendasarinya selesai. Neptune menggunakan penulisan kuorum yang membuat enam salinan data Anda di tiga Availability Zones, dan empat dari enam node penyimpanan harus mengakui penulisan untuknya agar dianggap tahan lama. Sementara utas kueri menunggu kuorum ini dari volume penyimpanan, ia terhenti, yang mengurangi utilisasi CPU.

Jika Anda memiliki pemuatan tulis serial di mana Anda melakukan satu penulisan demi satu penulisan dan menunggu yang pertama selesai sebelum memulai berikutnya, Anda dapat mengharapkan utilisasi CPU menjadi lebih rendah lagi. Jumlah pastinya akan menjadi fungsi dari jumlah v CPUs dan utas kueri (semakin banyak utas kueri, semakin sedikit keseluruhan CPU per kueri), dengan beberapa pengurangan yang disebabkan oleh menunggu I/O.

Untuk informasi selengkapnya tentang cara terbaik untuk mengukur instans DB, lihatMemilih jenis instans untuk Amazon Neptunus. Untuk harga setiap jenis instans, silakan lihat halaman harga Neptunus.

Pemantauan performa instans DB di Neptune

Anda dapat menggunakan CloudWatch metrik di Neptunus untuk memantau kinerja instans DB Anda dan melacak latensi kueri seperti yang diamati oleh klien. Lihat Menggunakan CloudWatch untuk memantau kinerja instans DB di Neptunus.

Penyimpanan, keandalan, dan ketersediaan Amazon Neptunus

Amazon Neptunus menggunakan arsitektur penyimpanan terdistribusi dan bersama yang diskalakan secara otomatis seiring dengan bertambahnya kebutuhan penyimpanan database Anda.

Data Neptunus disimpan dalam volume cluster, yang merupakan volume virtual tunggal yang menggunakan drive berbasis SSD Non-Volatile Memory Express (). NVMe Volume cluster terdiri dari kumpulan blok logis yang dikenal sebagai segmen. Masing-masing segmen ini dialokasikan 10 gigabyte (GB) penyimpanan. Data di setiap segmen direplikasi menjadi enam salinan, yang kemudian dialokasikan di tiga zona ketersediaan (AZs) di AWS Wilayah tempat cluster DB berada.

Ketika cluster DB Neptunus dibuat, ia dialokasikan satu segmen 10 GB. Ketika volume data meningkat dan melebihi penyimpanan yang saat ini dialokasikan, Neptunus secara otomatis memperluas volume cluster dengan menambahkan segmen baru. Volume cluster Neptunus dapat tumbuh hingga ukuran maksimum 128 tebibytes (TiB) di semua wilayah yang didukung kecuali Tiongkok GovCloud dan, di mana ia dibatasi hingga 64 TiB. NamunRilis: 1.0.2.2 (2020-03-09), untuk rilis mesin lebih awal dari, ukuran volume cluster dibatasi hingga 64 TiB di semua wilayah.

Volume cluster DB berisi semua data pengguna, indeks, dan kamus Anda (dijelaskan di Model Data Grafik Neptune bagian ini, serta metadata internal seperti log transaksi internal. Semua data grafik ini, termasuk indeks dan log internal, tidak dapat melebihi ukuran maksimum volume cluster.

I/O — Opsi penyimpanan yang dioptimalkan

Neptunus menawarkan dua model harga untuk penyimpanan:

- Penyimpanan standar — Penyimpanan standar menyediakan penyimpanan database yang hemat biaya untuk aplikasi dengan penggunaan sedang hingga rendah I/O .
- I/O — Penyimpanan yang dioptimalkan — Dengan I/O–Optimized storage, you pay only for the storage you are using, at a higher cost than for standard storage, and you pay nothing for the I/O itu Anda gunakan.

  I/O–Optimized storage is designed to meet the needs of I/O–intensive graph workloads at a predictable cost, with low I/O latency and consistent I/Othroughput.

  Untuk informasi selengkapnya, lihat I/O—Penyimpanan yang dioptimalkan.

Alokasi penyimpanan Neptunus

Meskipun volume cluster Neptunus dapat tumbuh hingga 128 TiB (atau 64 TiB di beberapa wilayah), Anda hanya dikenakan biaya untuk ruang yang sebenarnya dialokasikan. Total ruang yang dialokasikan ditentukan oleh tanda penyimpanan air tinggi, yang merupakan jumlah maksimum yang dialokasikan untuk volume cluster setiap saat selama keberadaannya.

Ini berarti bahwa bahkan jika data pengguna dihapus dari volume cluster, seperti oleh drop query sepertig`.V().drop()`, total ruang yang dialokasikan tetap sama. Neptunus secara otomatis mengoptimalkan ruang yang dialokasikan yang tidak terpakai untuk digunakan kembali di masa depan.

Selain data pengguna, dua jenis konten tambahan mengkonsumsi ruang penyimpanan internal, yaitu data kamus dan log transaksi internal. Meskipun data kamus disimpan dengan data grafik, itu tetap ada tanpa batas waktu, bahkan ketika data grafik yang didukungnya telah dihapus, yang berarti bahwa entri dapat digunakan kembali jika data diperkenalkan kembali. Data log internal disimpan di ruang penyimpanan internal terpisah yang memiliki tanda airnya sendiri yang tinggi. Ketika log internal kedaluwarsa, penyimpanan yang ditempati dapat digunakan kembali untuk log lain, tetapi tidak untuk data grafik. Jumlah ruang internal yang telah dialokasikan untuk log termasuk dalam total ruang yang dilaporkan oleh `VolumeBytesUsed` CloudWatch metrik.

Periksa cara Praktik terbaik penyimpanan untuk menjaga penyimpanan yang dialokasikan seminimal mungkin dan untuk menggunakan kembali ruang.

Penagihan penyimpanan Neptunus

Biaya penyimpanan ditagih berdasarkan tanda air penyimpanan yang tinggi, seperti dijelaskan di atas. Meskipun data Anda direplikasi menjadi enam salinan, Anda hanya ditagih untuk satu salinan data.

Anda dapat menentukan berapa tanda air tinggi penyimpanan saat ini dari cluster DB Anda dengan memantau `VolumeBytesUsed` CloudWatch metrik (lihatMemantau Neptunus Menggunakan Amazon CloudWatch).

Faktor lain yang dapat memengaruhi biaya penyimpanan Neptunus Anda termasuk snapshot database dan cadangan, yang ditagih secara terpisah sebagai penyimpanan cadangan dan didasarkan pada biaya penyimpanan Neptunus (lihat. CloudWatch metrik yang berguna untuk mengelola penyimpanan cadangan Neptunus

Namun, jika Anda membuat klon database Anda, klon menunjuk ke volume cluster yang sama dengan yang digunakan cluster DB Anda sendiri, sehingga tidak ada biaya penyimpanan tambahan untuk data asli. Perubahan selanjutnya pada klon menggunakan copy-on-write protokol, dan menghasilkan biaya penyimpanan tambahan.

Untuk informasi selengkapnya tentang harga Neptune, lihat Harga Amazon Neptune.

Praktik terbaik penyimpanan Neptunus

Karena jenis data tertentu mengkonsumsi penyimpanan permanen di Neptunus, gunakan praktik terbaik ini untuk menghindari lonjakan besar dalam pertumbuhan penyimpanan:

- Saat mendesain model data grafik Anda, hindari sebanyak mungkin menggunakan kunci properti dan nilai yang dihadapi pengguna yang bersifat sementara.
- Jika Anda berencana membuat perubahan pada model data Anda, jangan memuat data ke cluster DB yang ada menggunakan model baru sampai Anda menghapus data di cluster DB tersebut menggunakan API reset cepat. Hal terbaik adalah sering memuat data yang menggunakan model baru ke cluster DB baru.
- Transaksi yang beroperasi pada sejumlah besar data menghasilkan log internal yang besar, yang secara permanen dapat meningkatkan tanda air yang tinggi dari ruang log internal. Misalnya, satu transaksi yang menghapus semua data di cluster DB Anda dapat menghasilkan log internal besar yang akan membutuhkan mengalokasikan banyak penyimpanan internal dan dengan demikian secara permanen mengurangi ruang yang tersedia untuk data grafik.

  Untuk menghindari hal ini, bagi transaksi besar menjadi yang lebih kecil dan berikan waktu di antara mereka sehingga log internal terkait memiliki kesempatan untuk kedaluwarsa dan melepaskan penyimpanan internal mereka untuk digunakan kembali oleh log berikutnya.
- Untuk memantau pertumbuhan volume cluster Neptunus Anda, Anda dapat mengatur alarm pada CloudWatch metrik. `VolumeBytesUsed` CloudWatch Ini bisa sangat membantu jika data Anda mencapai ukuran maksimum volume cluster. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch alarm Amazon.

Satu-satunya cara untuk mengecilkan ruang penyimpanan yang digunakan oleh cluster DB Anda ketika Anda memiliki sejumlah besar ruang yang dialokasikan yang tidak terpakai adalah dengan mengekspor semua data dalam grafik Anda dan kemudian memuatnya kembali ke cluster DB baru. Lihat layanan dan utilitas ekspor data Neptunus untuk cara mudah mengekspor data dari cluster DB, dan pemuat massal Neptunus untuk cara mudah mengimpor data kembali ke Neptunus.

> **Note**
>
> Membuat dan memulihkan snapshot tidak mengurangi jumlah penyimpanan yang dialokasikan untuk cluster DB Anda, karena snapshot mempertahankan gambar asli penyimpanan dasar cluster. Jika sejumlah besar penyimpanan yang Anda alokasikan tidak digunakan, satu-satunya cara untuk mengecilkan jumlah penyimpanan yang dialokasikan adalah dengan mengekspor data grafik Anda dan memuatnya kembali ke cluster DB baru.

Keandalan penyimpanan Neptunus dan ketersediaan tinggi

Amazon Neptune dirancang agar andal, tahan lama, dan toleran terhadap kesalahan.

Fakta bahwa enam salinan data Neptunus Anda dipertahankan di tiga zona ketersediaan AZs () memastikan bahwa penyimpanan data sangat tahan lama, dengan kemungkinan kehilangan data yang sangat rendah. Data direplikasi secara otomatis di seluruh zona ketersediaan terlepas dari apakah ada instans DB di dalamnya, dan jumlah replikasi tidak tergantung pada jumlah instans DB di cluster Anda.

Ini berarti Anda dapat menambahkan replika baca dengan cepat, karena Neptunus tidak membuat salinan baru dari data grafik. Sebagai gantinya, replika baca terhubung ke volume cluster yang sudah berisi data Anda. Demikian pula, menghapus replika baca tidak menghapus data yang mendasarinya.

Anda dapat menghapus volume cluster dan datanya hanya setelah menghapus semua instance DB-nya.

Neptunus juga secara otomatis mendeteksi kegagalan pada segmen yang membentuk volume cluster. Ketika salinan data dalam segmen rusak, Neptunus segera memperbaiki segmen itu, menggunakan salinan data lain dalam segmen yang sama untuk memastikan bahwa data yang diperbaiki adalah saat ini. Akibatnya, Neptunus menghindari kehilangan data dan mengurangi kebutuhan untuk point-in-time melakukan pemulihan untuk memulihkan dari kegagalan disk.

Menghubungkan ke Titik Akhir Amazon Neptune.

Amazon Neptune menggunakan suatu klaster dari beberapa instans DB, bukan instans tunggal. Setiap koneksi Neptune ditangani oleh instans DB tertentu. Saat Anda terhubung ke klaster Neptune, nama dan port host yang Anda tentukan menunjuk ke handler intermediate yang disebut sebagai tiitk akhir. Endpoint adalah URL yang berisi alamat host dan port. Titik akhir Neptunus menggunakan koneksi Transport Layer terenkripsi). Security/Secure Sockets Layer (TLS/SSL

Neptune menggunakan mekanisme titik akhir untuk mengabstraksi koneksi ini sehingga Anda tidak perlu melakukan hardcode nama host, atau menulis logika Anda sendiri untuk koneksi rerouting saat beberapa instans DB tidak tersedia.

Dengan menggunakan titik akhir, Anda dapat memetakan setiap koneksi ke instans yang sesuai atau sekelompok instans bergantung pada kepentingan kasus penggunaan Anda. Endpoint kustom memungkinkan Anda terhubung ke subnet dari instans DB. Titik akhir berikut tersedia di klaster DB Neptune:

Titik akhir cluster Neptunus

Titik akhir klaster adalah titik akhir untuk klaster DB Neptune yang terhubung ke instans DB utama saat ini untuk klaster DB tersebut. Setiap klaster DB Neptune memiliki satu titik akhir klaster dan satu instans DB utama.

Titik akhir cluster menyediakan dukungan failover untuk read/write koneksi ke cluster DB. Gunakan titik akhir klaster untuk semua operasi tulis pada klaster DB tersebut, termasuk sisipan, pembaruan, penghapusan, dan perubahan DDL. Anda juga dapat menggunakan titik akhirklaster untuk operasi baca, seperti kueri.

Jika instans DB utama saat ini dari suatu klaster DB gagal, Neptune secara otomatis melakukan failover ke instans DB utama yang baru. Selama failover, klaster DB tersebut terus melayani permintaan koneksi ke titik akhirklaster dari instans DB utama yang baru, dengan interupsi layanan yang minimal.

Contoh berikut mengilustrasikan titik akhir klaster untuk suatu klaster DB Neptune.

```
mydbcluster.cluster-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Titik akhir pembaca Neptunus

Reader endpoint adalah titik akhir untuk klaster DB Neptune yang terhubung ke salah satu replika Neptune yang tersedia untuk klaster DB tersebut. Setiap klaster DB Neptune memiliki satu reader endpoint. Jika ada lebih dari satu replika Neptune, reader endpoint mengarahkan setiap permintaan koneksi ke salah satu replika Neptune.

Reader endpoint memberikan round-robin routing untuk koneksi baca saja ke klaster DB. Gunakan reader endpoint untuk operasi membaca, seperti kueri.

Anda tidak dapat menggunakan reader endpoint untuk operasi tulis kecuali Anda memiliki klaster single-instans (klaster yang tidak punya read-replika). Dalam hal tersebut dan kasus itu saja, pembaca dapat digunakan untuk operasi tullis demikian juga operasi baca.

Perutean round-robin reader endpoint bekerja dengan mengubah host yang ditunjuk entri DNS. Setiap kali Anda menyelesaikan DNS, Anda mendapatkan IP yang berbeda, dan koneksi dibuka terhadapnya. IPs Setelah sambungan dibuat, semua permintaan untuk koneksi tersebut dikirim ke host yang sama. Klien harus membuat sambungan baru dan menyelesaikan catatan DNS lagi untuk mendapatkan sambungan ke replika baca baru yang berbeda yang berpotensi.

> **Note**
>
> WebSockets Koneksi sering tetap hidup untuk waktu yang lama. Untuk mendapatkan replika baca yang berbeda, lakukan hal berikut:
>
> - Pastikan bahwa klien Anda menyelesaikan entri DNS setiap kali terhubung.
> - Tutup koneksi dan sambungkan kembali.

Berbagai perangkat lunak klien mungkin menyelesaikan DNS dengan cara yang berbeda. Misalnya, jika klien Anda menyelesaikan DNS dan kemudian menggunakan IP untuk setiap sambungan, itu mengarahkan semua permintaan ke host tunggal.

DNS caching untuk klien atau proxy menyelesaikan nama DNS ke titik akhir yang sama dari cache. Ini adalah masalah baik untuk routing robin routing maupun failover skenario.

> **Note**
>
> Menonaktifkan pengaturan caching DNS untuk memaksa resolusi DNS setiap waktu.

Klaster DB mendistribusikan permintaan koneksi ke reader endpoint di antara replika Neptune yang tersedia. Jika klaster DB hanya berisi satu instans utama reader endpoint melayani permintaan koneksi dari instans utama. Jika replika Neptune dibuat untuk klaster DB tersebut, reader endpoint terus melayani permintaan koneksi ke reader endpoint dari replika Neptune baru, dengan gangguan minimal dalam layanan.

Contoh berikut mengilustrasikan reader endpoint untuk suatu klaster DB Neptune.

```
mydbcluster.cluster-ro-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Titik akhir contoh Neptunus

Sebuah titik akhir instans adalah titik akhir untuk instans DB di klaster DB Neptune yang menghubungkan ke instans DB tertentu. Setiap instans DB dalam sebuah klaster DB, apa pun jenis instansnya, memiliki titik akhir instans uniknya sendiri. Jadi, ada satu titik akhir instans untuk instans DB utama saat ini dari klaster DB. Ada juga satu titik akhir instans untuk masing-masing replika Neptune di klaster DB.

Endpoint instans menyediakan kontrol langsung atas koneksi ke klaster DB, untuk skenario di mana menggunakan titik akhir klaster atau reader endpoint mungkin tidak sesuai. Misalnya, aplikasi klien Anda mungkin membutuhkan penyeimbang beban yang lebih halus berdasarkan jenis beban kerja. Dalam hal ini, Anda dapat mengkonfigurasi beberapa klien untuk melakukan koneksi ke Replika Neptune yang berbeda dalam klaster DB untuk mendistribusikan beban kerja baca.

Contoh berikut mengilustrasikan titik akhir instans untuk sebuah instans DB di suatu klaster DB Neptune.

```
mydbinstance.123456789012.us-east-1.neptune.amazonaws.com:8182
```

Titik akhir kustom Neptunus

Titik akhir kustom untuk klaster Neptune mewakili serangkaian instans DB yang Anda pilih. Saat Anda melakukan koneksi ke endpoint, Neptune memilih salah satu instans di grup untuk menangani koneksi tersebut. Anda menentukan instans mana yang dirujuk oleh endpoint, dan Anda memutuskan apa tujuan dari titik akhirtersebut.

Sebuah klaster DB Neptune tidak memiliki titik akhir kustom sampai Anda membuat satu, dan Anda dapat membuat hingga lima titik akhir kustom untuk setiap klaster Neptune.

Endpoint kustom menyediakan koneksi database load-balanced berdasarkan kriteria selain read-only atau read/write kemampuan instans DB. Karena koneksi dapat menuju ke setiap instans DB yang

dikaitkan dengan titik akhir, pastikan bahwa semua instans yang berada dalam grup tersebut memiliki karakteristik kapasitas memori dan performa serupa. Saat Anda menggunakan titik akhir kustom, Anda biasanya tidak menggunakan titik akhir pembaca untuk klaster tersebut.

Fitur ini ditujukan bagi pengguna tingkat lanjut dengan beban kerja khusus dimana tidak praktis untuk menyimpan semua Replika Neptune di identikal klaster. Dengan titik akhir kustom, Anda dapat menyesuaikan kapasitas instans DB yang digunakan untuk setiap koneksi.

Misalnya, jika Anda menentukan beberapa titik akhir kustom yang terhubung ke grup instans dengan kelas instans yang berbeda, Anda kemudian dapat mengarahkan pengguna dengan kebutuhan performa yang berbeda untuk titik akhir yang paling sesuai dengan kasus penggunaan mereka.

Contoh berikut mengilustrasikan titik akhir kustom untuk suatu instans DB di sebuah klaster DB Neptune.

```
myendpoint.cluster-custom-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Untuk informasi selengkapnya, lihat Bekerja dengan titik akhir khusus.

Pertimbangan titik akhir Neptune

Pertimbangkan hal berikut saat bekerja dengan titik akhir Neptune:

- Sebelum menggunakan titik akhir instans untuk terhubung ke instans DB tertentu dalam sebuah klaster DB, pertimbangkan untuk menggunakan titik akhir klaster atau reader endpoint untuk klaster DB sebagai gantinya.

  Titik akhir klaster dan reader endpoint menyediakan dukungan untuk skenario ketersediaan tinggi. Jika instans DB utama dari suatu klaster DB gagal, Neptune secara otomatis melakukan failover ke instans DB utama yang baru. Hal tersebut dilakukan dengan mempromosikan Replika Neptune yang sudah ada ke instans DB utama yang baru, atau membuat instans DB utama yang baru. Jika terjadi failover, Anda dapat menggunakan titik akhir klaster untuk melakukan koneksi kembali ke instans DB utama yang baru dipromosikan atau dibuat, atau menggunakan reader endpoint untuk melakukan koneksi kembali ke salah satu Replika Neptune di klaster DB.

  Jika Anda tidak mengambil pendekatan ini, Anda masih dapat memastikan bahwa Anda menghubungkan ke instans DB tepat di klaster DB untuk operasi dimaksudkan. Untuk melakukannya, Anda dapat secara manual atau terprogram menemukan serangkaian instans DB yang tersedia dalam klaster DB dan mengkonfirmasi jenis instans–nya setelah failover, sebelum menggunakan titik akhir instans dari instans DB tertentu.

Untuk informasi selengkapnya tentang failover, lihat Toleransi kesalahan untuk cluster DB Neptunus.

- Reader endpoint hanya mengarahkan koneksi ke replika Neptune yang tersedia di klaster DB Neptune. Ini tidak mengarahkan kueri tertentu.

  > **Important**
  >
  > Neptune tidak memuat keseimbangan.

  Jika Anda ingin memuat kueri seimbang untuk mendistribusikan beban kerja baca untuk suatu klaster DB, Anda harus mengelolanya dalam aplikasi Anda. Anda harus menggunakan titik akhir instans untuk menghubungkan langsung ke replika Neptune untuk menyeimbangkan beban.

- Perutean round-robin reader endpoint bekerja dengan mengubah host yang ditunjuk entri DNS. Klien harus membuat sambungan baru dan menyelesaikan catatan DNS lagi untuk mendapatkan sambungan ke replika baca baru yang berpotensi.

- Selama failover, reader endpoint mungkin akan mengarahkan koneksi ke instans DB utama yang baru dari klaster DB untuk waktu yang singkat saat replika Neptune dipromosikan menjadi instans DB utama yang baru.

Bekerja dengan titik akhir khusus di Neptunus

Saat Anda menambahkan instans DB ke titik akhir kustom atau menghapusnya dari titik akhir kustom, koneksi apa pun yang ada ke instans DB tersebut tetap aktif.

Anda dapat menetapkan daftar instans DB untuk disertakan dalam titik akhir kustom (daftar statis), atau satu untuk dikeculaikan dari titik akhir kustom (daftar pengecualian). Anda dapat menggunakan inclusion/exclusion mekanisme untuk membagi instance DB ke dalam grup dan memastikan bahwa titik akhir kustom mencakup semua instance DB di cluster. Setiap titik akhir kustom hanya dapat berisi salah satu jenis daftar tersebut.

Dalam Konsol Manajemen AWS, pilihan diwakili oleh kotak centang Lampirkan instance future ditambahkan ke cluster ini. Jika Anda membiarkan kotak centang kosong, titik akhirkustom menggunakan daftar statis yang hanya berisi instans DB yang ditentukan dalam dialog. Saat Anda mencentang kotak centang, titik akhir kustom menggunakan daftar pengecualian. Dalam hal ini, titik akhir kustom mewakili semua instans DB di dalam klaster (termasuk yang Anda tambahkan di masa mendatang) kecuali yang tidak dipilih dalam dialog.

Neptune tidak mengubah instans DB yang ditentukan dalam daftar statis atau pengecualian ketika instans DB mengubah peran antara instans utama dan Replika Neptune karena failover atau promosi.

Anda dapat mengaitkan instans DB dengan lebih dari satu titik akhirkustom. Misalnya, anggaplah Anda menambahkan instans DB baru ke sebuah klaster. Dalam hal instans DB ditambahkan ke semua titik akhir kustom yang memenuhi syarat. Daftar statis atau pengecualian didefinisikan untuk penentunya yang bisa ditambahkan instans DB.

Jika titik akhir mencakup daftar statis instans DB, Replika Neptune yang baru tidak ditambahkan ke dalamnya. Sebaliknya, jika titik akhir tersebut memiliki daftar pengecualian, Replika Neptune yang baru akan ditambahkan ke dalamnya asalkan tidak disebutkan dalam daftar pengecualian tersebut.

Jika Replika Neptune menjadi tidak tersedia, itu akan tetap dikaitkan dengan titik akhir kustomnya. Hal ini berlaku apakah tidak sehat, dihentikan, melakukan boot ulang, atau tidak tersedia karena alasan lain. Namun, selama itu tetap tidak tersedia, Anda tidak dapat menghubungkannya melewati titik akhir apapun.

Karena klaster Neptune yang baru dibuat tidak memiliki titik akhir kustom, Anda harus membuat dan mengelolanya sendiri. Hal ini juga berlaku untuk klaster Neptune yang dipulihkan dari snapshot, karena titik akhir kustom tidak termasuk dalam snapshot. Anda membuatnya kembali setelah memulihkan, dan memilih nama titik akhir baru jika klaster yang dipulihkan berada di wilayah yang sama dengan yang asli.

Membuat titik akhir kustom

Kelola titik akhir kustom menggunakan konsol Neptune. Lakukan ini dengan menavigasi ke halaman rincian untuk klaster Neptune Anda dan gunakan kontrol di bagian Titik akhir Kustom.

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Arahkan ke halaman detail klaster.

3. Pilih tindakan `Create custom endpoint` dalam bagian Titik akhir.
4. Pilih nama untuk titik akhir kustom yang unik untuk ID pengguna dan wilayah Anda. Nama harus 63 karakter atau kurang dan mengikuti bentuk berikut:

   endpointName.cluster-custom-*customerDnsIdentifier*.*dnsSuffix*

   Karena nama titik akhirkustom tidak menyertakan nama klaster Anda, Anda tidak perlu mengubah nama tersebut jika Anda mengganti nama klaster. Namun, Anda tidak dapat menggunakan kembali nama titik akhir kustom yang sama untuk lebih dari satu klaster di wilayah yang sama. Berikan setiap titik akhirkustom nama yang unik di seluruh klaster yang dimiliki oleh ID pengguna Anda dalam region tertentu.
5. Untuk memilih daftar instans DB yang tetap sama bahkan saat ukuran klaster bertambah, kosongkan kotak centang Lampirkan instans masa depan yang ditambahkan ke klaster ini. Saat kotak centang tersebut dicentang, titik akhir kustom akan secara dinamis menambahkan semua instans baru saat ditambahkan ke klaster.

Melihat titik akhir kustom

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Arahkan ke halaman detail klaster DB Anda.
3. Bagian Titik akhir hanya berisi informasi tentang titik akhir kustom (detail tentang titik akhir built-in tercantum dalam bagian Detail). Untuk melihat detail titik akhir kustom tertentu, pilih namanya untuk menampilkan halaman detail untuk titik akhir tersebut.

Mengubah titik akhir kustom

Anda dapat mengubah properti titik akhir kustom untuk mengubah instans DB mana yang dikaitkan dengan titik akhir tersebut. Anda juga dapat mengubah di antara daftar statis dan daftar pengecualian.

Anda tidak dapat terkoneksi ke atau menggunakan titik akhirkustom saat perubahan dari tindakan mengubah sedang berlangsung. Mungkin dibutuhkan beberapa menit setelah Anda membuat perubahan sebelum status titik akhi rkembali ke Tersedia dan Anda dapat menghubungkan kembali.

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/

2. Arahkan ke halaman detail klaster.
3. Di bagianTitik akhir, pilih nama titik akhir kustom yang ingin Anda edit.
4. Pada halaman detail untuk titik akhir itu, pilih tindakan Edit.

Menghapus titik akhir kustom

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Arahkan ke halaman detail klaster.
3. Di bagian Titik akhir, pilih nama titik akhir kustom yang ingin Anda hapus.
4. Pada halaman detail untuk titik akhir itu, pilih tindakan Hapus.

Mode Lab Neptune

Anda dapat menggunakan mode lab Amazon Neptune untuk mengaktifkan fitur baru yang ada di rilis mesin Neptune saat ini, tetapi belum siap untuk penggunaan produksi dan tidak diaktifkan secara default. Hal ini memungkinkan Anda mencoba fitur ini di lingkungan pengembangan dan pengujian Anda.

Menggunakan Mode Lab Neptune

Gunakan parameter cluster `neptune_lab_mode` DB untuk mengaktifkan atau menonaktifkan fitur. Anda melakukan hal ini dengan menyertakan *`(feature name)`*`=enabled` atau *`(feature name)`*`=disabled` di nilai dari parameter `neptune_lab_mode` dalam grup parameter Klaster DB.

Sebagai contoh, dalam rilis mesin ini Anda mungkin mengatur parameter `neptune_lab_mode` ke `Streams=disabled, ReadWriteConflictDetection=enabled`.

Untuk informasi tentang cara mengedit grup parameter klaster DB untuk database Anda, lihat Mengedit Grup Parameter. Perhatikan bahwa Anda tidak dapat mengedit grup parameter klaster DB default; jika Anda menggunakan grup default, Anda harus membuat grup parameter klaster DB baru sebelum Anda dapat mengatur parameter `neptune_lab_mode`.

> **Note**
>
> Ketika Anda membuat perubahan ke parameter cluster DB statis seperti`neptune_lab_mode`, Anda harus memulai kembali instance utama (penulis) cluster agar perubahan diterapkan. SebelumnyaRilis: 1.2.0.0 (2022-07-21), semua replika baca di cluster DB kemudian akan secara otomatis di-boot ulang ketika instance utama dimulai ulang. Dimulai denganRilis: 1.2.0.0 (2022-07-21), memulai ulang instance utama tidak menyebabkan replika apa pun dimulai ulang. Ini berarti Anda harus memulai ulang setiap instance secara terpisah untuk mengambil perubahan parameter cluster DB (lihatGrup parameter).

> **Important**
>
> Saat ini, jika Anda menyediakan parameter mode lab yang salah atau permintaan Anda gagal karena alasan lain, Anda mungkin tidak diberi tahu tentang kegagalan tersebut. Anda harus selalu memverifikasi bahwa permintaan perubahan mode lab telah berhasil dengan memanggil API status seperti ditunjukkan di bawah ini:

```
curl -G https://your-neptune-endpoint:port/status
```

Hasil status mencakup informasi mode lab yang akan menunjukkan apakah perubahan yang Anda minta dibuat atau tidak:

```
{
  "status":"healthy",
  "startTime":"Wed Dec 29 02:29:24 UTC 2021",
  "dbEngineVersion":"development",
  "role":"writer",
  "dfeQueryEngine":"viaQueryHint",
  "gremlin":{"version":"tinkerpop-3.5.2"},
  "sparql":{"version":"sparql-1.1"},
  "opencypher":{"version":"Neptune-9.0.20190305-1.0"},
  "labMode":{
    "ObjectIndex":"disabled",
    "ReadWriteConflictDetection":"enabled"
  },
  "features":{
    "LookupCache":{"status":"Available"},
    "ResultCache":{"status":"disabled"},
    "IAMAuthentication":"disabled",
    "Streams":"disabled",
    "AuditLog":"disabled"
  },
  "settings":{"clusterQueryTimeoutInMs":"120000"}
}
```

Fitur berikut saat ini diakses menggunakan mode lab:

Indeks OSGP

Neptune sekarang dapat mempertahankan indeks keempat, yaitu indeks OSGP, yang berguna untuk kumpulan data yang memiliki sejumlah besar predikat (lihat Mengaktifkan Indeks OSGP).

Anda dapat mengaktifkan indeks OSGP di klaster Neptune DB baru yang kosong dengan menetapkan `ObjectIndex=enabled` di parameter klaster DB `neptune_lab_mode`. Indeks OSGP hanya dapat diaktifkan di cluster DB baru yang kosong.

Secara default, indeks OSGP dinonaktifkan.

> **Note**
>
> Setelah mengatur parameter cluster `neptune_lab_mode` DB untuk mengaktifkan indeks OSGP, Anda harus memulai ulang instance penulis cluster agar perubahan diterapkan.

> **Warning**
>
> Jika Anda menonaktifkan indeks OSGP yang diaktifkan dengan menyetel `ObjectIndex=disabled` dan kemudian mengaktifkannya kembali setelah menambahkan lebih banyak data, indeks tidak akan dibangun dengan benar. Pembangunan kembali indeks sesuai permintaan tidak didukung, jadi Anda hanya boleh mengaktifkan indeks OSGP saat database kosong.

Mengaktifkan pengumpulan sampah kamus

Pengumpulan sampah kamus dapat diaktifkan untuk data grafik properti saat neptune-stream tidak diaktifkan melalui parameter. `DictionaryGCMode` Konkurensi dapat dikontrol melalui `DictionaryGCConcurrency` parameter. Lihat Pengumpulan sampah kamus untuk detail selengkapnya.

Semantik Transaksi yang Diformalisasi

Neptune telah memperbarui semantik formal untuk transaksi bersamaan (lihat Semantik Transaksi di Neptune).

Gunakan `ReadWriteConflictDetection` sebagai nama dalam parameter `neptune_lab_mode` yang mengaktifkan atau menonaktifkan semantik transaksi yang diformalisasi.

Secara default, semantik transaksi diformalisasi sudah diaktifkan. Jika Anda ingin kembali ke perilaku sebelumnya, sertakan `ReadWriteConflictDetection=disabled` dalam nilai yang ditetapkan untuk parameter `neptune_lab_mode` Klaster DB.

Dukungan datetime yang diperpanjang

Neptunus telah memperluas dukungan untuk fungsionalitas datetime. Untuk mengaktifkan datetime dengan format yang diperluas, sertakan `DatetimeMillisecond=enabled` dalam nilai yang ditetapkan untuk parameter DB Cluster`neptune_lab_mode`.

StrictTimeoutValidation

> **Note**
>
> Fitur ini tersedia mulai dari rilis mesin Neptunus 1.3.2.0.

Nilai default: diaktifkan (dinonaktifkan secara default sebelum rilis mesin Neptunus 1.4.0.0)

Nilai yang diizinkan: diaktifkan/dinonaktifkan

Jika parameter ini`enabled`, nilai batas waktu per kueri yang ditentukan sebagai opsi permintaan atau petunjuk kueri tidak dapat melebihi nilai yang ditetapkan secara global dalam pengaturan grup parameter. `neptune_query_timeout` Jika batas waktu per kueri melebihi pengaturan global, Neptunus melempar file. `InvalidParameterException` Dalam versi engine sebelum 1.4.0.0, parameter ini secara `disabled` default dan harus diaktifkan secara eksplisit.

Pengaturan ini dapat dikonfirmasi dalam respons pada `/status` titik akhir saat nilainya. `disabled`

Untuk informasi selengkapnya, lihat Batas waktu per kueri.

QRCMemoryEstimasi Akurat

> **Note**
>
> Fitur ini tersedia mulai dari rilis mesin Neptunus 1.4.0.0.

Nilai default: dinonaktifkan

Nilai yang diizinkan: diaktifkan/dinonaktifkan

Cache hasil kueri Gremlin, ketika diaktifkan, memungkinkan caching hasil kueri pada database. Dengan tuli perkiraan perkiraan digunakan untuk menentukan ukuran hasil yang di-cache, dengan

parameter mode lab ini `AccurateQRCMemoryEstimation` diaktifkan, estimasi ukuran untuk hasil yang di-cache akan menggunakan perkiraan ukuran yang akurat alih-alih perkiraan. Parameter labmode ini tersedia mulai dari rilis mesin Neptunus versi 1.4.0.0.

Mesin kueri alternatif Amazon Neptune (DFE)

Amazon Neptunus memiliki mesin kueri alternatif yang dikenal sebagai DFE yang menggunakan sumber daya instans DB seperti inti CPU, memori, I/O dan lebih efisien daripada mesin Neptunus asli.

> **Note**
>
> Dengan kumpulan data yang besar, mesin DFE mungkin tidak berjalan dengan baik pada instans t3.

Mesin DFE menjalankan kueri SPARQL, Gremlin dan OpenCypher, dan mendukung berbagai jenis paket, termasuk yang kiri dalam, lebat, dan hibrida. Operator paket dapat memanggil kedua operasi komputasi, yang berjalan pada serangkaian inti komputasi yang dicadangkan, dan I/O operasi, yang masing-masing berjalan pada utasnya sendiri di kumpulan utas. I/O

DFE menggunakan statistik yang dibuat sebelumnya tentang data grafik Neptune Anda untuk membuat keputusan yang tepat tentang bagaimana menyusun kueri. Lihat Statistik DFE untuk informasi tentang bagaimana statistik ini dihasilkan.

Pilihan jenis rencana dan jumlah utas komputasi yang digunakan dibuat secara otomatis berdasarkan statistik yang dihasilkan sebelumnya dan sumber daya yang tersedia di node kepala Neptune. Urutan hasil tidak ditentukan sebelumnya untuk rencana yang memiliki paralelisme komputasi internal.

Mengontrol di mana mesin Neptunus DFE digunakan

Secara default, parameter neptune_dfe_query_engine instance dari sebuah instance disetel `keviaQueryHint`, yang menyebabkan mesin DFE hanya digunakan untuk kueri OpenCypher dan untuk kueri Gremlin dan SPARQL yang secara eksplisit menyertakan petunjuk kueri yang disetel ke. `useDFE true`

Anda dapat sepenuhnya mengaktifkan mesin DFE sehingga digunakan sedapat mungkin dengan menyetel parameter `neptune_dfe_query_engine` instance `keenabled`.

Anda juga dapat menonaktifkan DFE dengan menyertakan petunjuk `useDFE` kueri untuk kueri Gremlin tertentu atau kueri SPARQL. Petunjuk kueri ini memungkinkan Anda mencegah DFE mengeksekusi kueri tertentu.

Anda dapat menentukan apakah DFE diaktifkan dalam sebuah instance menggunakan Status instans panggilan, seperti ini:

```
curl -G https://your-neptune-endpoint:port/status
```

Respons status kemudian menentukan apakah DFE diaktifkan atau tidak:

```
{
  "status":"healthy",
  "startTime":"Wed Dec 29 02:29:24 UTC 2021",
  "dbEngineVersion":"development",
  "role":"writer",
  "dfeQueryEngine":"viaQueryHint",
  "gremlin":{"version":"tinkerpop-3.5.2"},
  "sparql":{"version":"sparql-1.1"},
  "opencypher":{"version":"Neptune-9.0.20190305-1.0"},
  "labMode":{
    "ObjectIndex":"disabled",
    "ReadWriteConflictDetection":"enabled"
  },
  "features":{
    "ResultCache":{"status":"disabled"},
    "IAMAuthentication":"disabled",
    "Streams":"disabled",
    "AuditLog":"disabled"
  },
  "settings":{"clusterQueryTimeoutInMs":"120000"}
}
```

Hasil `explain` dan `profile` Gremlin memberi tahu Anda apakah kueri sedang dieksekusi oleh DFE. Lihat Informasi yang terkandung dalam laporan `explain` Gremlin untuk `explain` dan Laporan `profile` DFE untuk `profile`.

Demikian pula, `explain` SPARQL memberi tahu Anda apakah kueri SPARQL sedang dieksekusi oleh DHE. Untuk detail selengkapnya, lihat Contoh output `explain` SPARQL saat DFE diaktifkan dan Operator `DFENode`.

Konstruksi kueri yang didukung oleh Neptune DFE

Saat ini, Neptune DFE mendukung subset dari kontruksi kueri SPARQL dan Gremlin.

Untuk SPARQL, ini adalah bagian dari pola grafik dasar konjungtif.

Untuk Gremlin, umumnya subset dari query yang berisi rantai traversal yang tidak mengandung beberapa langkah yang lebih kompleks.

Anda dapat mengetahui apakah salah satu kueri Anda sedang dieksekusi secara keseluruhan atau sebagian oleh DFE sebagai berikut:

- Di Gremlin, hasil `explain` dan `profile` memberi tahu Anda bagian apa dari kueri yang sedang dieksekusi oleh DFE, jika ada. Lihat Informasi yang terkandung dalam laporan `explain` Gremlin untuk `explain` dan Laporan `profile` DFE untuk `profile`. Lihat juga Menyetel kueri Gremlin menggunakan `explain` dan `profile`.

  Detail tentang dukungan mesin Neptune untuk langkah-langkah Gremlin individual didokumentasikan dalam Dukungan langkah Gremlin.
- Demikian pula, `explain` SPARQL memberi tahu Anda apakah kueri SPARQL sedang dieksekusi oleh DHE. Untuk detail selengkapnya, lihat Contoh output `explain` SPARQL saat DFE diaktifkan dan Operator `DFENode`.

Mengelola statistik untuk Neptune DFE yang akan digunakan

> **Note**
>
> Support untuk OpenCypher tergantung pada mesin query DFE di Neptunus.
> Mesin DFE pertama kali tersedia dalam mode lab di rilis mesin Neptunus 1.0.3.0, dan mulai di rilis mesin Neptunus 1.0.5.0, menjadi diaktifkan secara default, tetapi hanya untuk digunakan dengan petunjuk kueri dan untuk dukungan OpenCypher.
> Dimulai dengan rilis mesin Neptunus 1.1.1.0, mesin DFE tidak lagi dalam mode lab, dan sekarang dikontrol menggunakan neptune_dfe_query_engine parameter instance dalam grup parameter DB instans.

Mesin DFE menggunakan informasi tentang data dalam grafik Neptunus Anda untuk membuat trade-off yang efektif saat merencanakan eksekusi kueri. Informasi ini mengambil bentuk statistik yang mencakup apa yang disebut set karakteristik dan statistik predikat yang dapat memandu perencanaan kueri.

Dimulai dengan rilis engine 1.2.1.0, Anda dapat mengambil informasi ringkasan tentang grafik Anda dari statistik ini menggunakan GetGraphSummaryAPI atau titik akhir. `summary`

Statistik DFE ini saat ini dihasilkan kembali setiap kali lebih dari 10% data dalam grafik Anda telah berubah atau ketika statistik terbaru berusia lebih dari 10 hari. Namun, pemicu ini dapat berubah di masa depan.

> **Note**
>
> Pembuatan statistik dinonaktifkan `T3` dan `T4g` instance karena dapat melebihi kapasitas memori dari jenis instance tersebut.

Anda dapat mengelola pembuatan statistik DFE melalui salah satu titik akhir berikut:

- `https://`*`your-neptune-host`*`:`*`port`*`/rdf/statistics` (untuk SPARQL).
- `https://`*`your-neptune-host`*`:`*`port`*`/propertygraph/statistics`(untuk Gremlin dan OpenCypher), dan versi alternatifnya:. `https://`*`your-neptune-host`*`:`*`port`*`/pg/statistics`

> **Note**
>
> Pada rilis mesin 1.1.1.0, titik akhir statistik Gremlin (`https://your-neptune-host:port/gremlin/statistics`) tidak digunakan lagi demi titik akhir atau. `propertygraph` pg Ini masih didukung untuk kompatibilitas mundur tetapi dapat dihapus di rilis mendatang.
> Pada rilis mesin 1.2.1.0, titik akhir statistik SPARQL (`https://your-neptune-host:port/sparql/statistics`) tidak digunakan lagi demi titik akhir. `rdf` Ini masih didukung untuk kompatibilitas mundur tetapi dapat dihapus di rilis mendatang.

Dalam contoh di bawah ini, `$STATISTICS_ENDPOINT` singkatan dari salah satu titik akhir URLs ini.

> **Note**
>
> Jika titik akhir statistik DFE berada pada instans pembaca, satu-satunya permintaan yang dapat diproses adalah permintaan status. Permintaan lainnya akan gagal dengan `ReadOnlyViolationException`.

Batas ukuran untuk pembuatan statistik DFE

Saat ini, pembuatan statistik DFE berhenti jika salah satu dari batas ukuran berikut tercapai:

- Jumlah set karakteristik yang dihasilkan tidak boleh melebihi 50.000.
- Jumlah statistik predikat yang dihasilkan tidak boleh melebihi satu juta.

Batas ini bisa berubah.

Status statistik DFE saat ini

Anda dapat memeriksa status statistik DFE saat ini menggunakan permintaan `curl`:

```
curl -G "$STATISTICS_ENDPOINT"
```

Respons ke permintaan status berisi bidang-bidang berikut:

- `status` — Kode pengembalian HTTP dari permintaan. Jika permintaan berhasil, kodenya adalah `200`. Lihat Kesalahan umum untuk daftar kesalahan umum.

- `payload`:
  - `autoCompute`— (Boolean) Menunjukkan apakah pembuatan statistik otomatis diaktifkan atau tidak.
  - `active`— (Boolean) Menunjukkan apakah pembuatan statistik DFE diaktifkan atau tidak.
  - `statisticsId` — Laporan ID dari pembuatan statistik saat ini yang dijalankan. Nilai `-1` menunjukkan bahwa tidak ada statistik yang dihasilkan.
  - `date`— Waktu UTC di mana statistik DFE baru-baru ini dihasilkan, dalam format ISO 8601.

    > **Note**
    >
    > Sebelum rilis mesin 1.2.1.0, ini diwakili dengan presisi menit, tetapi dari rilis mesin 1.2.1.0 ke depan, ini diwakili dengan presisi milidetik (misalnya,). `2023-01-24T00:47:43.319Z`

  - `note`— Catatan tentang masalah dalam kasus di mana statistik tidak valid.
  - `signatureInfo`— Berisi informasi tentang set karakteristik yang dihasilkan dalam statistik (sebelum rilis mesin 1.2.1.0, bidang ini diberi nama`summary`). Ini umumnya tidak langsung ditindaklanjuti:
    - `signatureCount`— Jumlah total tanda tangan di semua set karakteristik.
    - `instanceCount`— Jumlah total instans karakteristik-set.
    - `predicateCount`— Jumlah total predikat unik.

Respons terhadap permintaan status ketika tidak ada statistik telah dihasilkan terlihat seperti ini:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : -1
   }
}
```

Jika statistik DFE tersedia, responsnya terlihat seperti ini:

```
{
```

```
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : true,
    "statisticsId" : 1588893232718,
    "date" : "2020-05-07T23:13Z",
    "summary" : {
      "signatureCount" : 5,
      "instanceCount" : 1000,
      "predicateCount" : 20
    }
  }
}
```

Jika pembuatan statistik DFE gagal, misalnya karena melebihi batas ukuran statistik, responnya terlihat seperti ini:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : 1588713528304,
    "date" : "2020-05-05T21:18Z",
    "note" : "Limit reached: Statistics are not available"
  }
}
```

Menonaktifkan pembuatan otomatis statistik DFE

Secara default, pembuatan otomatis statistik DFE diaktifkan saat Anda mengaktifkan DFE.

Anda dapat menonaktifkan pembuatan otomatis sebagai berikut:

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "disableAutoCompute" }'
```

Jika permintaan berhasil, kode respons HTTP-nya adalah `200` dan responsnya adalah:

```
{
  "status" : "200 OK"
}
```

Anda dapat mengonfirmasi bahwa pembuatan otomatis dinonaktifkan dengan mengeluarkan permintaan status dan memeriksa bahwa bidang `autoCompute` di dalam respons diatur ke `false`.

Menonaktifkan pembuatan statistik otomatis tidak mengakhiri komputer statistik yang sedang berlangsung.

Jika Anda membuat permintaan untuk menonaktifkan pembuatan otomatis instans pembaca daripada instans penulis klaster DB Anda, permintaan gagal dengan kode kembali HTTP 400 dan output seperti berikut:

```
{
  "detailedMessage" : "Writes are not permitted on a read replica instance",
  "code" : "ReadOnlyViolationException",
  "requestId":"8eb8d3e5-0996-4a1b-616a-74e0ec32d5f7"
}
```

Lihat Kesalahan umum untuk daftar kesalahan umum lainnya.

Mengaktifkan kembali pembuatan otomatis statistik DFE

Secara default, pembuatan otomatis statistik DFE sudah diaktifkan saat Anda mengaktifkan DFE. Jika Anda menonaktifkan pembuatan otomatis, Anda dapat mengaktifkannya kembali nanti sebagai berikut:

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "enableAutoCompute" }'
```

Jika permintaan berhasil, kode respons HTTP-nya adalah `200` dan responsnya adalah:

```
{
  "status" : "200 OK"
}
```

Anda dapat mengonfirmasi bahwa pembuatan otomatis diaktifkan dengan mengeluarkan permintaan status dan memeriksa bahwa bidang `autoCompute` di dalam respons diatur ke `true`.

Memicu pembuatan statistik DFE secara manual

Anda dapat memulai pembuatan statistik DFE secara manual sebagai berikut:

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "refresh" }'
```

Jika permintaan berhasil, outputnya adalah sebagai berikut, dengan kode kembali HTTP 200:

```
{
  "status" : "200 OK",
  "payload" : {
    "statisticsId" : 1588893232718
  }
}
```

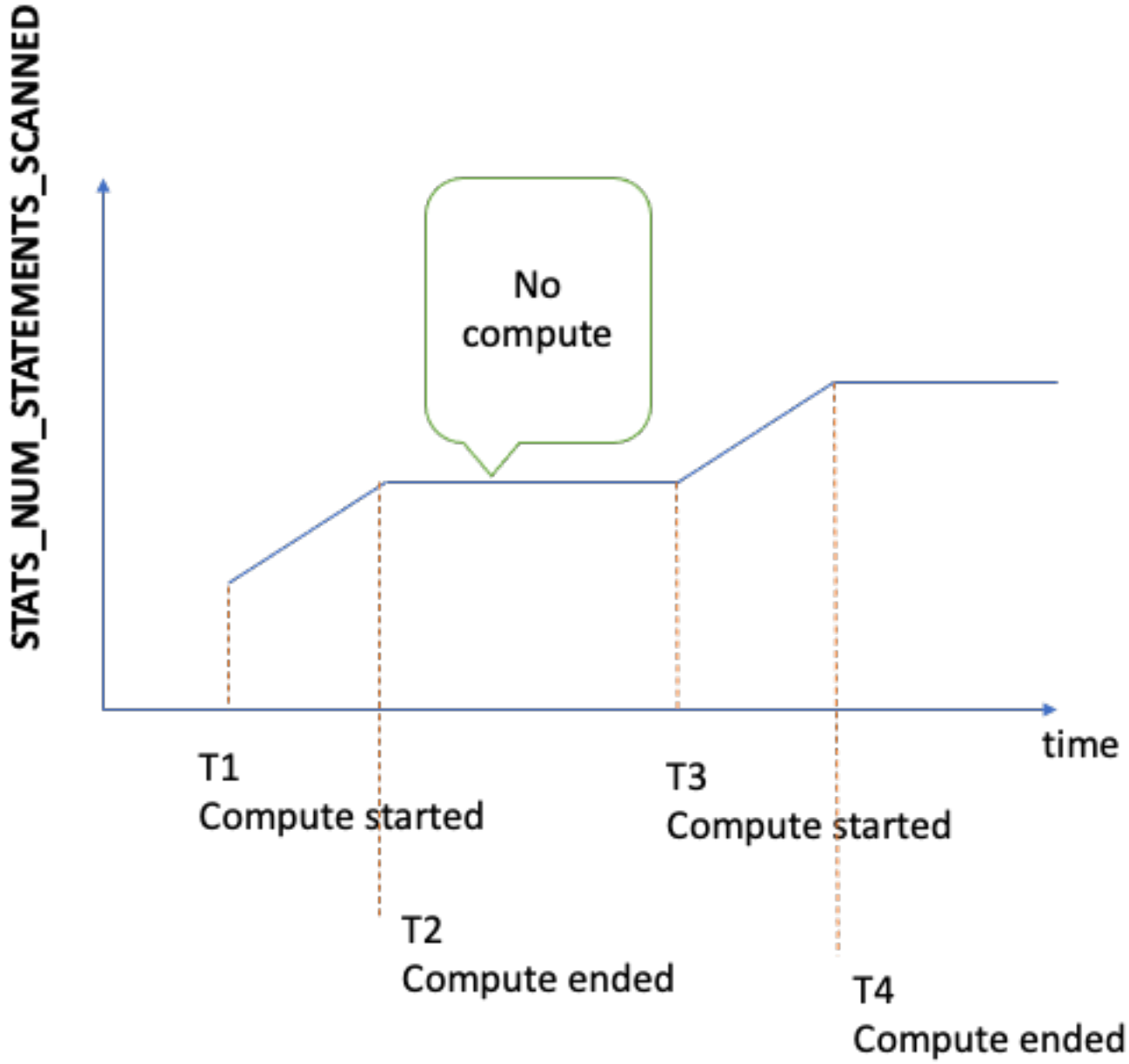
`statisticsId` dalam output adalah ID dari proses pembuatan statistik yang saat ini terjadi. Jika proses sudah berjalan pada saat permintaan, permintaan mengembalikan ID yang dijalankan daripada memulai yang baru. Hanya satu proses pembuatan statistik yang dapat terjadi pada satu waktu.

Jika fail-over terjadi sementara statistik DFE sedang dihasilkan, node penulis baru akan mengambil pos pemeriksaan yang diproses terakhir dan melanjutkan proses statistik dari sana.

Menggunakan **`StatsNumStatementsScanned`** CloudWatch metrik untuk memantau perhitungan statistik

`StatsNumStatementsScanned` CloudWatch Metrik mengembalikan jumlah total pernyataan yang dipindai untuk perhitungan statistik sejak server dimulai. Ini diperbarui pada setiap irisan perhitungan statistik.

Setiap kali perhitungan statistik dipicu, angka ini meningkat, dan ketika tidak ada perhitungan yang terjadi, itu tetap konstan. Oleh karena itu, melihat sebidang `StatsNumStatementsScanned` nilai dari waktu ke waktu memberi Anda gambaran yang cukup jelas tentang kapan perhitungan statistik terjadi dan seberapa cepat:



Saat perhitungan terjadi, kemiringan grafik menunjukkan seberapa cepat (semakin curam kemiringan, semakin cepat statistik dihitung).

Jika grafik hanyalah garis datar di 0, fitur statistik telah diaktifkan, tetapi tidak ada statistik yang dihitung sama sekali. Jika fitur statistik telah dinonaktifkan, atau jika Anda menggunakan versi mesin yang tidak mendukung perhitungan statistik, fitur tersebut `StatsNumStatementsScanned` tidak ada.

Seperti disebutkan sebelumnya, Anda dapat menonaktifkan perhitungan statistik menggunakan API statistik, tetapi membiarkannya mati dapat mengakibatkan statistik tidak mutakhir, yang pada gilirannya dapat mengakibatkan pembuatan rencana kueri yang buruk untuk mesin DFE.

Lihat Memantau Neptunus Menggunakan Amazon CloudWatch untuk informasi tentang cara menggunakan CloudWatch.

Menggunakan otentikasi AWS Identity and Access Management (IAM) dengan titik akhir statistik DFE

Anda dapat mengakses titik akhir statistik DFE secara aman dengan otentikasi IAM dengan menggunakan awscurl atau alat lain yang bekerja dengan HTTPS dan IAM. Lihat Menggunakan `awscurl` dengan kredensyal sementara untuk terhubung dengan aman ke cluster DB dengan otentikasi IAM diaktifkan untuk melihat cara mengatur kredensil yang tepat. Setelah Anda selesai melakukannya, Anda kemudian dapat membuat permintaan status seperti ini:

```
awscurl "$STATISTICS_ENDPOINT" \
    --region (your region) \
    --service neptune-db
```

Atau, misalnya, Anda dapat membuat file JSON dengan nama `request.json` yang berisi:

```
{ "mode" : "refresh" }
```

Anda kemudian dapat secara manual memulai pembuatan statistik seperti ini:

```
awscurl "$STATISTICS_ENDPOINT" \
    --region (your region) \
    --service neptune-db \
    -X POST -d @request.json
```

Menghapus statistik DFE

Anda dapat menghapus semua statistik dalam database dengan membuat permintaan HTTP DELETE ke titik akhir statistik:

```
curl -X "DELETE" "$STATISTICS_ENDPOINT"
```

Kode pengembalian HTTP yang valid adalah:

- `200`— penghapusan berhasil.

  Dalam hal ini, respons tipikal akan terlihat seperti:

```
{
  "status" : "200 OK",
  "payload" : {
      "active" : false,
      "statisticsId" : -1
  }
}
```

- `204`Tidak ada statistik yang harus dihapus.

  Dalam hal ini, responsnya kosong (tidak ada respons).

Jika Anda mengirim permintaan penghapusan ke titik akhir statistik pada node pembaca, a `ReadOnlyViolationException` dilemparkan.

Kode kesalahan umum untuk permintaan statistik DFE

Berikut ini adalah daftar kesalahan umum yang dapat terjadi ketika Anda membuat permintaan ke titik akhir statistik:

- `AccessDeniedException` — Kode kembali: `400`. Pesan: `Missing Authentication Token`.
- `BadRequestException`(untuk Gremlin dan OpenCypher) - Kode pengembalian:. `400` Pesan: `Bad route: /pg/statistics`.
- `BadRequestException`(untuk data RDF) — Kode pengembalian:. `400` Pesan: `Bad route: /rdf/statistics`.
- `InvalidParameterException` — Kode kembali: `400`. Pesan: `Statistics command parameter 'mode' has unsupported value '`*`the invalid value`*`'`.
- `MissingParameterException` — Kode kembali: `400`. Pesan: `Content-type header not specified.`.
- `ReadOnlyViolationException` — Kode kembali: `400`. Pesan: `Writes are not permitted on a read replica instance`.

Misalnya, jika Anda membuat permintaan ketika DFE dan statistik tidak diaktifkan, Anda akan mendapatkan respons seperti berikut:

```
{
  "code" : "BadRequestException",
  "requestId" : "b2b8f8ee-18f1-e164-49ea-836381a3e174",
  "detailedMessage" : "Bad route: /sparql/statistics"
}
```

Mendapatkan laporan ringkasan singkat tentang grafik Anda

API ringkasan grafik Neptunus mengambil informasi berikut tentang grafik Anda:

- Untuk grafik properti (PG), API ringkasan grafik mengembalikan daftar read-only label node dan edge dan kunci properti, bersama dengan jumlah node, tepi, dan properti.
- Untuk grafik kerangka deskripsi sumber daya (RDF), API ringkasan grafik mengembalikan daftar kelas dan kunci predikat hanya-baca, bersama dengan jumlah paha depan, subjek, dan predikat.

> **Note**
>
> API ringkasan grafik diperkenalkan dalam rilis mesin Neptunus 1.2.1.0.

Dengan API ringkasan grafik, Anda dapat dengan cepat mendapatkan pemahaman tingkat tinggi tentang ukuran dan konten data grafik Anda. Anda juga dapat menggunakan API secara interaktif dalam notebook Neptunus menggunakan sihir Neptune Workbench. `%summary` Dalam aplikasi grafik, API dapat digunakan untuk meningkatkan hasil pencarian dengan memberikan label node atau tepi yang ditemukan sebagai bagian dari pencarian.

Data ringkasan grafik diambil dari statistik DFE yang dihitung oleh mesin DFE Neptunus selama runtime, dan tersedia setiap kali statistik DFE tersedia. Statistik diaktifkan secara default saat Anda membuat cluster DB Neptunus baru.

> **Note**
>
> Pembuatan statistik dinonaktifkan `t3` dan tipe `t4` instance (yaitu, tipe on `db.t3.medium` dan `db.t4g.medium` instance) untuk menghemat memori. Akibatnya, data ringkasan grafik juga tidak tersedia pada jenis instance tersebut.

Anda dapat memeriksa status statistik DFE menggunakan API status statistik. Selama pembuatan statistik secara otomatis belum dinonaktifkan, statistik diperbarui secara otomatis secara berkala.

Jika Anda ingin memastikan bahwa statistik seterbaru mungkin saat Anda meminta ringkasan grafik, Anda dapat memicu pembaruan statistik secara manual tepat sebelum mengambil ringkasan. Jika grafik berubah saat statistik sedang dihitung, mereka tentu akan sedikit tertinggal, tetapi tidak banyak.

Menggunakan API ringkasan grafik untuk mengambil informasi ringkasan grafik

Untuk grafik properti yang Anda kueri menggunakan Gremlin atau OpenCypher, Anda dapat mengambil ringkasan grafik dari titik akhir ringkasan grafik properti. Ada URI panjang dan pendek untuk titik akhir ini:

- `https://your-neptune-host:port/propertygraph/statistics/summary`
- `https://your-neptune-host:port/pg/statistics/summary`

Untuk grafik RDF yang Anda kueri menggunakan SPARQL, Anda dapat mengambil ringkasan grafik dari titik akhir ringkasan RDF:

- `https://your-neptune-host:port/rdf/statistics/summary`

Titik akhir ini hanya-baca, dan hanya mendukung operasi HTTP. `GET` Jika $GRAPH_SUMMARY_ENDPOINT diatur ke alamat titik akhir mana pun yang ingin Anda kueri, Anda dapat mengambil data ringkasan menggunakan dan HTTP sebagai berikut: `curl GET`

```
curl -G "$GRAPH_SUMMARY_ENDPOINT"
```

Jika tidak ada statistik yang tersedia saat Anda mencoba mengambil ringkasan grafik, responsnya terlihat seperti ini:

```
{
  "detailedMessage": "Statistics are not available. Summary can only be generated after
 statistics are available.",
  "requestId": "48c1f788-f80b-b69c-d728-3f6df579a5f6",
  "code": "StatisticsNotAvailableException"
}
```

Parameter kueri **`mode`** URL untuk API ringkasan grafik

API ringkasan grafik menerima parameter kueri URL bernama`mode`, yang dapat mengambil salah satu dari dua nilai, yaitu `basic` (default) dan`detailed`. Untuk grafik RDF, respons ringkasan grafik `detailed` mode berisi bidang tambahan`subjectStructures`. Untuk grafik

properti, respons ringkasan grafik terperinci berisi dua bidang tambahan, yaitu `nodeStructures` dan`edgeStructures`.

Untuk meminta respons ringkasan `detailed` grafik, sertakan `mode` parameter sebagai berikut:

```
curl -G "$GRAPH_SUMMARY_ENDPOINT?mode=detailed"
```

Jika `mode` parameter tidak ada, `basic` mode digunakan secara default, jadi meskipun dimungkinkan untuk menentukan secara `?mode=basic` eksplisit, ini tidak perlu.

Respons ringkasan grafik untuk grafik properti (PG)

Untuk grafik properti kosong, respons ringkasan grafik terperinci terlihat seperti ini:

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-01-10T07:58:47.972Z",
    "graphSummary" : {
      "numNodes" : 0,
      "numEdges" : 0,
      "numNodeLabels" : 0,
      "numEdgeLabels" : 0,
      "nodeLabels" : [ ],
      "edgeLabels" : [ ],
      "numNodeProperties" : 0,
      "numEdgeProperties" : 0,
      "nodeProperties" : [ ],
      "edgeProperties" : [ ],
      "totalNodePropertyValues" : 0,
      "totalEdgePropertyValues" : 0,
      "nodeStructures" : [ ],
      "edgeStructures" : [ ]
    }
  }
}
```

Respons ringkasan grafik properti (PG) memiliki bidang berikut:

- **`status`**— kode pengembalian HTTP dari permintaan. Jika permintaan berhasil, kodenya adalah 200.

Lihat Kesalahan ringkasan grafik umum untuk daftar kesalahan umum.

- **payload**
  - **version**— Versi respons ringkasan grafik ini.
  - **lastStatisticsComputationTime** Stempel waktu, dalam format ISO 8601, saat Neptunus terakhir menghitung statistik.
  - **graphSummary**
    - **numNodes**— Jumlah node dalam grafik.
    - **numEdges**— Jumlah tepi dalam grafik.
    - **numNodeLabels**— Jumlah label node yang berbeda dalam grafik.
    - **numEdgeLabels**— Jumlah label tepi yang berbeda dalam grafik.
    - **nodeLabels**— Daftar label node yang berbeda dalam grafik.
    - **edgeLabels**— Daftar label tepi yang berbeda dalam grafik.
    - **numNodeProperties**— Jumlah properti simpul yang berbeda dalam grafik.
    - **numEdgeProperties**— Jumlah properti tepi yang berbeda dalam grafik.
    - **nodeProperties**— Daftar properti node yang berbeda dalam grafik, bersama dengan jumlah node di mana setiap properti digunakan.
    - **edgeProperties**— Daftar properti tepi yang berbeda dalam grafik bersama dengan jumlah tepi di mana setiap properti digunakan.
    - **totalNodePropertyValues**— Jumlah total penggunaan semua properti node.
    - **totalEdgePropertyValues**— Jumlah total penggunaan semua properti tepi.
    - **nodeStructures**- Bidang ini hanya *mode=detailed* ada ketika ditentukan dalam permintaan. Ini berisi daftar struktur simpul, yang masing-masing berisi bidang-bidang berikut:
      - **count**— Jumlah node yang memiliki struktur khusus ini.
      - **nodeProperties**— Daftar properti simpul yang ada dalam struktur khusus ini.
      - **distinctOutgoingEdgeLabels**— Daftar label tepi keluar yang berbeda hadir dalam struktur khusus ini.
    - **edgeStructures**- Bidang ini hanya *mode=detailed* ada ketika ditentukan dalam permintaan. Ini berisi daftar struktur tepi, yang masing-masing berisi bidang-bidang berikut:
      - **count**— Jumlah tepi yang memiliki struktur khusus ini.
      - **edgeProperties**— Daftar properti tepi yang ada dalam struktur khusus ini.

Respons ringkasan grafik untuk grafik RDF

Untuk grafik RDF kosong, respons ringkasan grafik terperinci terlihat seperti ini:

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-01-10T07:58:47.972Z",
    "graphSummary" : {
      "numDistinctSubjects" : 0,
      "numDistinctPredicates" : 0,
      "numQuads" : 0,
      "numClasses" : 0,
      "classes" : [ ],
      "predicates" : [ ],
      "subjectStructures" : [ ]
    }
  }
}
```

Respons ringkasan grafik RDF memiliki bidang-bidang berikut:

- **`status`**— kode pengembalian HTTP dari permintaan. Jika permintaan berhasil, kodenya adalah 200.

  Lihat Kesalahan ringkasan grafik umum untuk daftar kesalahan umum.
- **`payload`**
  - **`version`**— Versi respons ringkasan grafik ini.
  - **`lastStatisticsComputationTime`** Stempel waktu, dalam format ISO 8601, saat Neptunus terakhir menghitung statistik.
  - **`graphSummary`**
    - **`numDistinctSubjects`**— Jumlah subjek yang berbeda dalam grafik.
    - **`numDistinctPredicates`**— Jumlah predikat yang berbeda dalam grafik.
    - **`numQuads`**— Jumlah paha depan dalam grafik.
    - **`numClasses`**— Jumlah kelas dalam grafik.
    - **`classes`**— Daftar kelas dalam grafik.
    - **`predicates`**— Daftar predikat dalam grafik, bersama dengan jumlah predikat.

- **`subjectStructures`**- Bidang ini hanya *`mode=detailed`* ada ketika ditentukan dalam permintaan. Ini berisi daftar struktur subjek, yang masing-masing berisi bidang-bidang berikut:
  - **`count`**— Jumlah kemunculan struktur khusus ini.
  - **`predicates`**— Daftar predikat yang ada dalam struktur khusus ini.

Contoh respons ringkasan grafik properti (PG)

Berikut adalah respon ringkasan rinci untuk grafik properti yang berisi contoh dataset rute udara grafik properti:


```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-03-01T14:35:03.804Z",
    "graphSummary" : {
      "numNodes" : 3748,
      "numEdges" : 51300,
      "numNodeLabels" : 4,
      "numEdgeLabels" : 2,
      "nodeLabels" : [
        "continent",
        "country",
        "version",
        "airport"
      ],
      "edgeLabels" : [
        "contains",
        "route"
      ],
      "numNodeProperties" : 14,
      "numEdgeProperties" : 1,
      "nodeProperties" : [
        {
          "desc" : 3748
        },
        {
          "code" : 3748
        },
        {
          "type" : 3748
```

```
        },
        {
          "country" : 3503
        },
        {
          "longest" : 3503
        },
        {
          "city" : 3503
        },
        {
          "lon" : 3503
        },
        {
          "elev" : 3503
        },
        {
          "icao" : 3503
        },
        {
          "region" : 3503
        },
        {
          "runways" : 3503
        },
        {
          "lat" : 3503
        },
        {
          "date" : 1
        },
        {
          "author" : 1
        }
      ],
      "edgeProperties" : [
        {
          "dist" : 50532
        }
      ],
      "totalNodePropertyValues" : 42773,
      "totalEdgePropertyValues" : 50532,
      "nodeStructures" : [
        {
```

```
          "count" : 3471,
          "nodeProperties" : [
            "city",
            "code",
            "country",
            "desc",
            "elev",
            "icao",
            "lat",
            "lon",
            "longest",
            "region",
            "runways",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [
            "route"
          ]
        },
        {
          "count" : 161,
          "nodeProperties" : [
            "code",
            "desc",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [
            "contains"
          ]
        },
        {
          "count" : 83,
          "nodeProperties" : [
            "code",
            "desc",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [ ]
        },
        {
          "count" : 32,
          "nodeProperties" : [
            "city",
            "code",
```

```
            "country",
            "desc",
            "elev",
            "icao",
            "lat",
            "lon",
            "longest",
            "region",
            "runways",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [ ]
        },
        {
          "count" : 1,
          "nodeProperties" : [
            "author",
            "code",
            "date",
            "desc",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [ ]
        }
      ],
      "edgeStructures" : [
        {
          "count" : 50532,
          "edgeProperties" : [
            "dist"
          ]
        }
      ]
    }
  }
}
```

Contoh respons ringkasan grafik RDF

Berikut adalah respons ringkasan terperinci untuk grafik RDF yang berisi sampel dataset rute udara RDF:

```
{
```

```
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-03-01T14:54:13.903Z",
    "graphSummary" : {
      "numDistinctSubjects" : 54403,
      "numDistinctPredicates" : 19,
      "numQuads" : 158571,
      "numClasses" : 4,
      "classes" : [
        "http://kelvinlawrence.net/air-routes/class/Version",
        "http://kelvinlawrence.net/air-routes/class/Airport",
        "http://kelvinlawrence.net/air-routes/class/Continent",
        "http://kelvinlawrence.net/air-routes/class/Country"
      ],
      "predicates" : [
        {
          "http://kelvinlawrence.net/air-routes/objectProperty/route" : 50656
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/dist" : 50656
        },
        {
          "http://kelvinlawrence.net/air-routes/objectProperty/contains" : 7004
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/code" : 3747
        },
        {
          "http://www.w3.org/2000/01/rdf-schema#label" : 3747
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/type" : 3747
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/desc" : 3747
        },
        {
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" : 3747
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/icao" : 3502
        },
        {
```

```
        "http://kelvinlawrence.net/air-routes/datatypeProperty/lat" : 3502
      },
      {
        "http://kelvinlawrence.net/air-routes/datatypeProperty/region" : 3502
      },
      {
        "http://kelvinlawrence.net/air-routes/datatypeProperty/runways" : 3502
      },
      {
        "http://kelvinlawrence.net/air-routes/datatypeProperty/longest" : 3502
      },
      {
        "http://kelvinlawrence.net/air-routes/datatypeProperty/elev" : 3502
      },
      {
        "http://kelvinlawrence.net/air-routes/datatypeProperty/lon" : 3502
      },
      {
        "http://kelvinlawrence.net/air-routes/datatypeProperty/country" : 3502
      },
      {
        "http://kelvinlawrence.net/air-routes/datatypeProperty/city" : 3502
      },
      {
        "http://kelvinlawrence.net/air-routes/datatypeProperty/author" : 1
      },
      {
        "http://kelvinlawrence.net/air-routes/datatypeProperty/date" : 1
      }
    ],
    "subjectStructures" : [
      {
        "count" : 50656,
        "predicates" : [
          "http://kelvinlawrence.net/air-routes/datatypeProperty/dist"
        ]
      },
      {
        "count" : 3471,
        "predicates" : [
          "http://kelvinlawrence.net/air-routes/datatypeProperty/city",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/country",
          "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
```

```
            "http://kelvinlawrence.net/air-routes/datatypeProperty/elev",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/icao",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lat",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lon",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/longest",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/region",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/runways",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://kelvinlawrence.net/air-routes/objectProperty/route",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 238,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://kelvinlawrence.net/air-routes/objectProperty/contains",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 31,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/city",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/country",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/elev",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/icao",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lat",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lon",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/longest",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/region",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/runways",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
```

```
          "count" : 6,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 1,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/author",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/date",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        }
      ]
    }
  }
}
```

Menggunakan otentikasi AWS Identity and Access Management (IAM) dengan titik akhir ringkasan grafik

Anda dapat mengakses titik akhir ringkasan grafik secara aman dengan otentikasi IAM dengan menggunakan awscurl atau alat lain yang bekerja dengan HTTPS dan IAM. Lihat Menggunakan `awscurl` dengan kredensyal sementara untuk terhubung dengan aman ke cluster DB dengan otentikasi IAM diaktifkan untuk melihat cara mengatur kredensil yang tepat. Setelah Anda melakukannya, Anda kemudian dapat membuat permintaan seperti ini:

```
awscurl "$GRAPH_SUMMARY_ENDPOINT" \
    --region (your region) \
    --service neptune-db
```

> **Important**
>
> Identitas atau peran IAM yang menciptakan kredensyal sementara harus memiliki kebijakan IAM yang dilampirkan yang memungkinkan tindakan IAM. GetGraphSummary

Lihat Kesalahan Autentikasi IAM daftar kesalahan IAM umum yang mungkin Anda temui.

Kode kesalahan umum yang dapat dikembalikan oleh permintaan ringkasan grafik

Kode kesalahan layanan Neptunus	Statu HTT	Pesan	Skenario Kesalahan	Mitigasi
AccessDeniedException	403	Token Otentikasi Hilang.	Permintaan yang tidak ditandatangani atau ditandatangani salah dikirim ke database Neptunus dengan IAM diaktifkan.	Tanda tangani permintaan dengan SigV4 sebelum mengirim (lihatIAM dan ringkasan grafik).
	403	Pengguna: *(user ARN)* tidak berwenang untuk melakukan: neptune-db: pada sumber daya:GetGraphSummary . *(resource ARN)*	Kebijakan IAM tidak mengizinkan tindakan GetGraphSummarysaat permintaan ringkasan grafik dikirim ke database Neptunus dengan IAM diaktifkan.	Pastikan bahwa kebijakan IAM yang dilampirkan pada pengguna atau peran yang membuat permintaan memungkinkan `GetGraphSummary` tindakan.
BadRequestException	400	Statistik dinonaktifkan, jadi ringkasan grafik juga dinonaktifkan.	Mencoba mengambil ringkasan pada jenis instance burstable (`t3ataut4g`) di	Gunakan jenis instans di mana pembuatan statistik diaktifkan (semua instance yang

Kode kesalahan layanan Neptunus	Statu HTT	Pesan	Skenario Kesalahan	Mitigasi
			mana statistik dinonaktifkan.	didukung kecuali `t3` dan`t4g`).
	400	Rute buruk: */rdf/statistics/summarypathapi*	Permintaan dikirim ke jalur yang tidak valid.	Gunakan rute yang benar untuk titik akhir ringkasan grafik.
InvalidParameterException	400	Permintaan berisi parameter yang tidak diketahui : *(unknown parameter or parameters)* ".	Ketika parameter yang tidak valid ditentukan dalam permintaan.	Hanya gunakan parameter yang valid (sepertimode) dalam permintaan.
InvalidParameterException	400	Parameter kueri URI 'mode' memiliki nilai '*(invalid value)*' yang tidak didukung.	Ketika parameter URL 'mode' dalam permintaan diikuti oleh nilai yang tidak valid.	Gunakan nilai yang valid (seperti `basic` ataudetailed) saat menentuka n parameter URL 'mode'.
MethodNotAllowedException	405	Metode Tidak Diizinkan.	Memanggil titik akhir ringkasan dengan metode HTTP apa pun selain `GET` (seperti `POST` atauDELETE).	Gunakan `GET` metode HTTP saat memanggil titik akhir ringkasan.

Kode kesalahan layanan Neptunus	Statu HTT	Pesan	Skenario Kesalahan	Mitigasi
StatisticsNotAvailableException	400	Statistik belum dihitung, ringkasan grafik akan tersedia setelah perhitungan statistik selesai.	Tidak ada statistik yang tersedia saat permintaan dikirim ke titik akhir ringkasan.	Tunggu sampai pembuatan statistik selesai. Anda dapat memeriksa status pembuatan statistik menggunakan API status statistik.
	400	Batas statistik tercapai, sehingga ringkasan grafik tidak tersedia.	Generasi statistik telah berhenti karena mencapai batas ukuran statistik.	Ringkasan grafik tidak tersedia pada grafik ini.

Misalnya, jika Anda membuat permintaan untuk membuat grafik titik akhir ringkasan dalam database Neptunus yang mengaktifkan autentikasi IAM, dan izin yang diperlukan tidak ada dalam kebijakan IAM pemohon, maka Anda akan mendapatkan respons seperti berikut:

```
{
  "detailedMessage": "User: arn:aws:iam::(account ID):(user or user name) is not
 authorized to perform: neptune-db:GetGraphSummary on resource: arn:aws:neptune-
db:(region):(account ID):(cluster resource ID)/*",
  "requestId": "7ac2b98e-b626-d239-1d05-74b4c88fce82",
  "code": "AccessDeniedException"
}
```

Konektivitas Amazon Neptunus JDBC

Amazon Neptune telah merilis driver JDBC open-source yang mendukung kueri OpenCypher, Gremlin, SQL-Gremlin, dan SPARQL. Konektivitas JDBC memudahkan untuk terhubung ke Neptunus dengan alat intelijen bisnis (BI) seperti Tableau. Tidak ada biaya tambahan untuk menggunakan driver JDBC dengan Neptunus — Anda masih membayar hanya untuk sumber daya Neptunus yang dikonsumsi.

Driver ini kompatibel dengan JDBC 4.2, dan membutuhkan setidaknya Java 8. Lihat dokumentasi JDBC API untuk informasi tentang cara menggunakan driver JDBC.

GitHub Proyek, tempat Anda dapat mengajukan masalah dan membuka permintaan fitur, berisi dokumentasi terperinci untuk driver:

Driver JDBC untuk Amazon Neptunus

- Menggunakan SQL dengan driver JDBC
- Menggunakan Gremlin dengan Driver JDBC
- Menggunakan OpenCypher dengan Driver JDBC
- Menggunakan SPARQL dengan Driver JDBC

Memulai dengan driver Neptunus JDBC

Untuk menggunakan driver Neptunus JDBC agar terhubung ke instans Neptunus, driver JDBC harus digunakan pada instans Amazon EC2 di VPC yang sama dengan cluster DB Neptunus Anda, atau instans harus tersedia melalui terowongan SSH atau penyeimbang beban. Terowongan SSH dapat diatur di driver secara internal, atau dapat diatur secara eksternal.

Anda dapat mengunduh drivernya di sini. Pengemudi dikemas sebagai file JAR tunggal dengan nama seperti`neptune-jdbc-1.0.0-all.jar`. Untuk menggunakannya, letakkan file JAR di `classpath` aplikasi Anda. Atau, jika aplikasi Anda menggunakan Maven atau Gradle, Anda dapat menggunakan perintah Maven atau Gradle yang sesuai untuk menginstal driver dari JAR.

Pengemudi memerlukan URL koneksi JDBC untuk terhubung dengan Neptunus, dalam bentuk seperti ini:

```
jdbc:neptune:(connection
 type)://(host);property=value;property=value;...;property=value
```

Bagian untuk setiap bahasa kueri dalam GitHub proyek menjelaskan properti yang dapat Anda atur dalam URL koneksi JDBC untuk bahasa kueri tersebut.

Jika file JAR ada di aplikasi Anda`classpath`, tidak ada konfigurasi lain yang diperlukan. Anda dapat menghubungkan driver menggunakan `DriverManager` antarmuka JDBC dan string koneksi Neptunus. Misalnya, jika cluster DB Neptunus Anda dapat diakses melalui `neptune-example.com` titik akhir pada port 8182, Anda akan dapat terhubung dengan OpenCypher seperti ini:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

void example() {
    String url = "jdbc:neptune:opencypher://bolt://neptune-example:8182";

    Connection connection = DriverManager.getConnection(url);
    Statement statement = connection.createStatement();

    connection.close();
}
```

Bagian dokumentasi dalam GitHub proyek untuk setiap bahasa kueri menjelaskan cara membuat string koneksi saat menggunakan bahasa kueri tersebut.

Menggunakan Tableau dengan driver Neptunus JDBC

Untuk menggunakan Tableau dengan driver Neptunus JDBC, mulailah dengan mengunduh dan menginstal versi terbaru Tableau Desktop. Unduh file JAR untuk driver Neptunus JDBC, dan juga file konektor Neptunus Tableau (file). `.taco`

Untuk terhubung ke Tableau untuk Neptunus di Mac

1. Tempatkan file JAR driver Neptunus JDBC di folder. `/Users/`*`(your user name)`*`/Library/Tableau/Drivers`
2. Tempatkan file `.taco` konektor Neptunus Tableau di folder. `/Users/`*`(your user name)`*`/Documents/My Tableau Repository/Connectors`
3. Jika autentikasi IAM diaktifkan, siapkan lingkungan untuk itu. Perhatikan bahwa variabel lingkungan diatur dalam`.zprofile/`, `.zshenv/.bash_profile`,, dan sebagainya, tidak akan bekerja. Variabel lingkungan harus diatur sehingga dapat dimuat oleh aplikasi GUI.

Salah satu cara untuk mengatur kredensyal Anda adalah dengan menempatkan kunci akses dan kunci rahasia Anda dalam file. `/Users/(your user name)/.aws/credentials`

Cara mudah untuk mengatur wilayah layanan adalah dengan membuka terminal dan memasukkan perintah berikut, menggunakan wilayah aplikasi Anda (misalnya,`us-east-1`):

```
launchctl setenv SERVICE_REGION region name
```

Ada cara lain untuk mengatur variabel lingkungan yang bertahan setelah restart, tetapi teknik apa pun yang Anda gunakan harus mengatur variabel yang dapat diakses oleh aplikasi GUI.

4. Untuk mendapatkan variabel lingkungan untuk dimuat ke GUI di Mac, masukkan perintah ini di terminal:

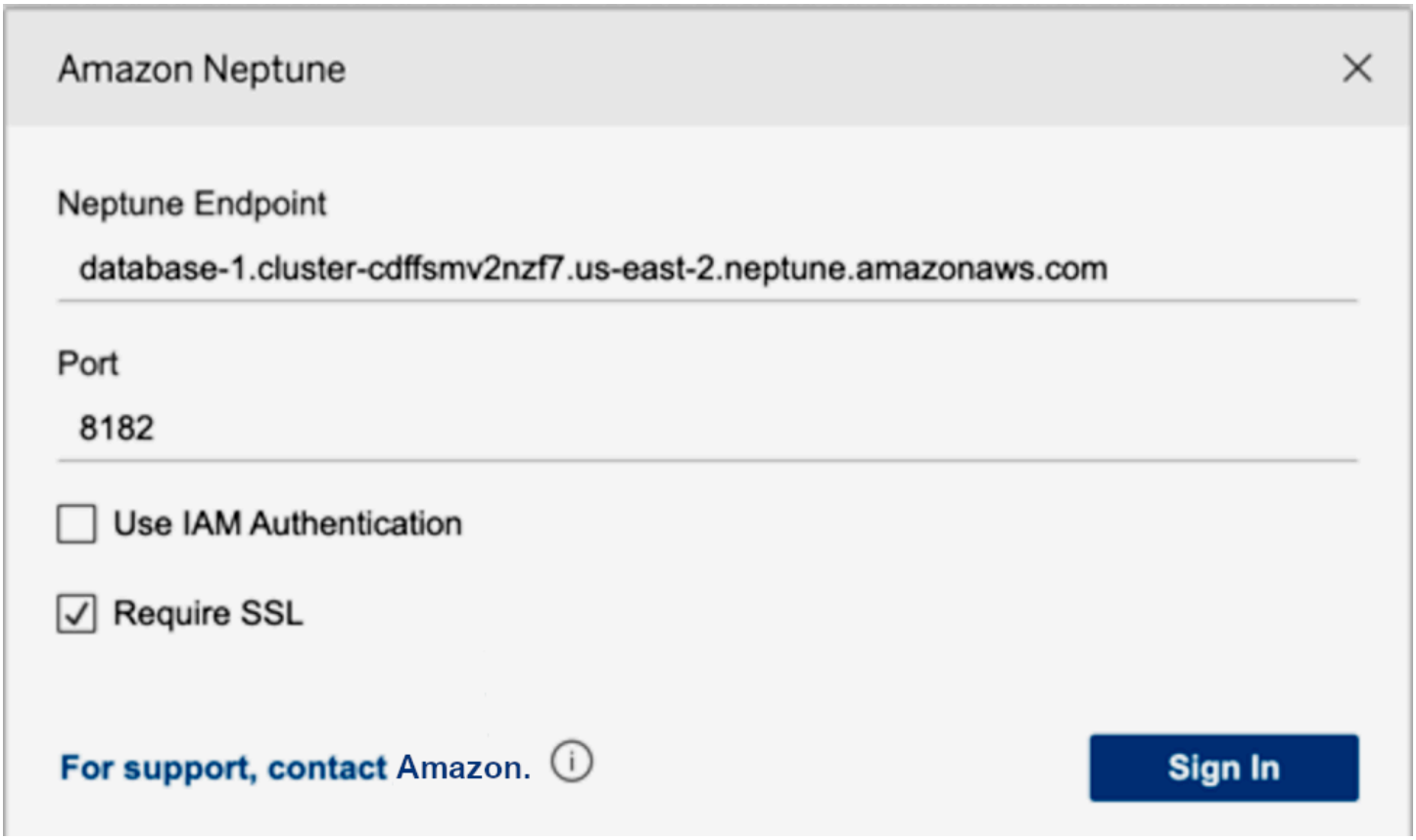
```
/Applications/Tableau/Desktop/2021.1.app/Contents/MacOS/Tableau
```

Untuk terhubung ke Tableau untuk Neptunus pada mesin Windows

1. Tempatkan file JAR driver Neptunus JDBC di folder. `C:\Program Files\Tableau\Drivers`
2. Tempatkan file `.taco` konektor Neptunus Tableau di folder. `C:\Users\(your user name)\Documents\My Tableau Repository\Connectors`
3. Jika autentikasi IAM diaktifkan, siapkan lingkungan untuk itu.

   Ini bisa sesederhana pengaturan penggunaACCESS_KEY,SECRET_KEY, dan variabel `SERVICE_REGION` lingkungan.

Dengan Tableau terbuka, pilih Lainnya di sisi kiri jendela. Jika file konektor Tableau berada dengan benar, Anda dapat memilih Amazon AWS Neptunus dengan dalam daftar yang muncul:



Anda tidak perlu mengedit port, atau menambahkan opsi koneksi apa pun. Masukkan titik akhir Neptunus Anda dan atur konfigurasi IAM dan SSL Anda (Anda harus mengaktifkan SSL jika Anda menggunakan IAM).

Saat Anda memilih Masuk, mungkin diperlukan waktu lebih dari 30 detik untuk terhubung jika grafik Anda besar. Tableau mengumpulkan tabel simpul dan tepi dan menggabungkan simpul di tepi, serta menciptakan visualisasi.

Memecahkan masalah koneksi driver JDBC

Jika driver gagal terhubung ke server, gunakan `isValid` fungsi `Connection` objek JDBC untuk memeriksa apakah koneksi valid. Jika fungsi kembali`false`, artinya koneksi tidak valid, periksa apakah titik akhir yang terhubung sudah benar dan Anda berada di VPC cluster DB Neptunus Anda atau Anda memiliki terowongan SSH yang valid ke cluster.

Jika Anda mendapat `No suitable driver found for` *`(connection string)`* tanggapan dari `DriverManager.getConnection` panggilan, kemungkinan ada masalah di awal string koneksi Anda. Pastikan string koneksi Anda dimulai seperti ini:

```
jdbc:neptune:opencypher://...
```

Untuk mengumpulkan informasi lebih lanjut tentang koneksi, Anda dapat menambahkan `LogLevel` ke string koneksi Anda seperti ini:

```
jdbc:neptune:opencypher://(JDBC URL):(port);logLevel=trace
```

Atau, Anda dapat menambahkan `properties.put("logLevel", "trace")` properti input Anda untuk mencatat informasi jejak.

Pembaruan mesin Amazon Neptunus

Amazon Neptune merilis update mesin secara berkala. Anda dapat menentukan versi rilis mesin yang telah Anda instal saat ini menggunakan API status instans.

Rilis mesin tercantum di Rilis mesin untuk Amazon Neptunus, dan patch dicantumkan di Pembaruan Teranyar.

Anda dapat menemukan informasi lebih lanjut tentang bagaimana pembaruan dirilis dan cara meningkatkan mesin Neptunus di database Anda di pemeliharaan Cluster. Misalnya, penomoran versi dijelaskan dalam nomor versi Engine.

Penanganan Pengecualian dan Mencoba Lagi

Membangun aplikasi yang kuat di Neptunus sering berarti mempersiapkan hal yang tidak terduga, terutama ketika menangani kesalahan yang dikembalikan oleh database. Salah satu tanggapan paling umum terhadap pengecualian sisi server adalah mencoba kembali operasi yang gagal. Meskipun logika coba lagi sangat penting untuk sistem yang tangguh, Anda perlu menyadari bahwa tidak semua kesalahan harus diperlakukan dengan cara yang sama. Daripada mengandalkan perilaku coba ulang generik, pendekatan yang bijaksana dapat membantu Anda membangun aplikasi yang lebih andal dan efisien.

Mengapa coba lagi logika penting

Logika coba lagi adalah komponen penting dari aplikasi terdistribusi apa pun. Masalah sementara seperti ketidakstabilan jaringan, kendala sumber daya sementara, atau konflik modifikasi bersamaan dapat menyebabkan operasi gagal. Dalam banyak kasus, kegagalan ini tidak menunjukkan masalah permanen dan dapat diselesaikan dengan menunggu dan mencoba lagi. Menerapkan strategi coba lagi yang solid mengakui realitas lingkungan yang tidak sempurna dalam sistem terdistribusi, memastikan keandalan dan kontinuitas yang lebih kuat dengan lebih sedikit kebutuhan untuk intervensi manual.

Risiko percobaan ulang tanpa pandang bulu

Mencoba kembali setiap kesalahan secara default dapat menyebabkan beberapa konsekuensi yang tidak diinginkan:

- Peningkatan pertengkaran — Ketika operasi yang gagal karena konkurensi tinggi dicoba berulang kali, pertengkaran keseluruhan bisa menjadi lebih buruk. Hal ini dapat mengakibatkan siklus transaksi yang gagal dan kinerja yang menurun.
- Kelelahan sumber daya — Percobaan ulang tanpa pandang bulu dapat mengkonsumsi sumber daya sistem tambahan, baik di sisi klien maupun server. Hal ini berpotensi menyebabkan pelambatan atau bahkan degradasi layanan.
- Peningkatan latensi untuk klien — Percobaan ulang yang berlebihan dapat menyebabkan penundaan yang signifikan untuk aplikasi klien, terutama jika setiap percobaan ulang melibatkan masa tunggu. Hal ini dapat berdampak negatif pada pengalaman pengguna dan proses hilir.

Mengembangkan strategi coba lagi yang praktis

Untuk membangun aplikasi yang tangguh dan efisien, kembangkan strategi coba ulang yang disesuaikan dengan kondisi kesalahan spesifik yang mungkin dihadapi aplikasi Anda. Berikut adalah beberapa pertimbangan untuk memandu pendekatan Anda:

- Identifikasi kesalahan yang dapat dicoba ulang - Tidak semua pengecualian harus dicoba lagi. Misalnya, kesalahan sintaks, kegagalan otentikasi, atau kueri yang tidak valid seharusnya tidak memicu percobaan ulang. Neptunus menyediakan kode kesalahan dan rekomendasi umum yang kesalahannya aman untuk dicoba lagi, tetapi Anda perlu menerapkan logika yang sesuai dengan kasus penggunaan Anda.
- Implementasikan backoff eksponensial — Untuk kesalahan sementara, gunakan strategi backoff eksponensial untuk secara progresif meningkatkan waktu tunggu di antara percobaan ulang. Ini membantu meringankan pertengkaran dan mengurangi risiko kegagalan cascading.
- Pertimbangkan panjang jeda awal — Melakukan percobaan ulang pertama terlalu cepat mungkin berakhir dengan kesalahan yang sama jika server belum diberi cukup waktu untuk merilis sumber daya yang dibutuhkan kueri agar berhasil. Jeda yang lebih lama dalam situasi yang tepat dapat mengurangi permintaan yang terbuang dan tekanan server.
- Tambahkan jitter ke backoff — Meskipun backoff eksponensial efektif, itu masih dapat menyebabkan badai coba lagi yang disinkronkan jika banyak klien gagal pada saat yang sama dan kemudian mencoba lagi bersama. Menambahkan jitter, variasi acak kecil ke penundaan backoff, membantu menyebarkan upaya coba lagi sehingga mengurangi kemungkinan semua klien mencoba lagi secara bersamaan dan menyebabkan lonjakan beban lainnya.
- Batasi upaya coba lagi — Tetapkan jumlah percobaan ulang maksimum yang wajar untuk mencegah loop tak terbatas dan kehabisan sumber daya.
- Pantau dan sesuaikan — Pantau terus tingkat kesalahan aplikasi Anda dan sesuaikan strategi coba ulang sesuai kebutuhan. Jika Anda melihat sejumlah besar percobaan ulang untuk operasi tertentu, pertimbangkan apakah operasi dapat dioptimalkan atau diserialkan.

Contoh alur perencanaan

Strategi coba lagi yang tepat tergantung pada sifat kegagalan, beban kerja, dan pola kesalahan yang Anda amati. Tabel berikut merangkum beberapa skenario kegagalan umum dan bagaimana pertimbangan strategi coba lagi berlaku untuk masing-masing. Paragraf penjelasan mengikuti untuk konteks tambahan.

Skenario	Dicoba ulang?	Backoff & Jitter	Jeda Awal	Coba lagi Batas	Pantau & Sesuaikan
CME sesekali pada Kueri Pendek	Ya	Backoff pendek, tambahkan jitter	Pendek (misalnya, 100ms)	Tinggi	Perhatikan kenaikan Tarif CME
CME yang Sering pada Kueri yang Berjalan Lebih Lama	Ya	Backoff yang lebih panjang, tambahkan jitter	Lebih lama (misalnya, 2s)	Sedang	Selidiki dan kurangi pertengkaran
Batas Memori pada Kueri Mahal	Ya	Backoff panjang	Panjang (misalnya, 5-10s)	Rendah	Optimalkan kueri, waspada jika persisten
Batas Waktu pada Kueri Sedang	Mungkin	Backoff moderat, tambahkan jitter	Sedang (misalnya, 1s)	Rendah hingga Sedang	Menilai beban server dan desain kueri

Skenario 1: CME sesekali pada kueri singkat

Untuk beban kerja yang jarang `ConcurrentModificationException` muncul selama pembaruan singkat dan sederhana, kesalahan ini biasanya bersifat sementara dan aman untuk dicoba lagi. Gunakan jeda awal singkat (misalnya, 100 milidetik) sebelum mencoba lagi pertama. Kali ini memungkinkan kunci singkat untuk dihapus. Gabungkan ini dengan backoff eksponensial pendek dan jitter untuk menghindari percobaan ulang yang disinkronkan. Karena biaya mencoba ulang rendah, batas coba lagi yang lebih tinggi masuk akal. Namun, pantau tingkat CME untuk menangkap tren apa pun terhadap peningkatan pertikaian dalam data Anda.

Skenario 2: CME yang sering terjadi pada kueri yang berjalan lama

Jika aplikasi Anda sering terlihat CMEs pada kueri yang berjalan lama, ini menunjukkan pertengkaran yang lebih parah. Dalam hal ini, mulailah dengan jeda awal yang lebih lama (misalnya, 2 detik), untuk memberikan kueri saat ini yang menahan kunci cukup waktu untuk diselesaikan. Gunakan backoff eksponensial yang lebih panjang dan tambahkan jitter. Batasi jumlah percobaan ulang untuk menghindari penundaan dan penggunaan sumber daya yang berlebihan. Jika pertengkaran berlanjut, tinjau beban kerja Anda untuk pola dan pertimbangkan untuk membuat serial pembaruan atau mengurangi konkurensi untuk mengatasi akar masalahnya.

Skenario 3: Batas memori pada kueri mahal

Ketika kesalahan berbasis memori terjadi selama kueri intensif sumber daya yang diketahui, percobaan ulang dapat masuk akal, tetapi hanya setelah jeda awal yang lama (misalnya, 5 hingga 10 detik atau lebih) untuk memungkinkan server melepaskan sumber daya. Gunakan strategi backoff panjang dan tetapkan batas coba ulang yang rendah, karena kegagalan berulang tidak mungkin diselesaikan tanpa perubahan pada kueri atau beban kerja. Kesalahan persisten harus memicu peringatan dan meminta peninjauan kompleksitas kueri dan penggunaan sumber daya.

Skenario 4: Batas waktu pada kueri moderat

Batas waktu pada kueri yang cukup mahal adalah kasus yang lebih ambigu. Terkadang, percobaan ulang mungkin berhasil jika batas waktu disebabkan oleh lonjakan sementara dalam beban server atau kondisi jaringan. Mulailah dengan jeda awal yang moderat (misalnya, 1 detik) untuk memberi sistem kesempatan untuk pulih. Terapkan backoff moderat dan tambahkan jitter untuk menghindari percobaan ulang yang disinkronkan. Pertahankan batas coba ulang rendah hingga sedang, karena batas waktu berulang mungkin menunjukkan masalah yang lebih dalam dengan kueri atau kapasitas server. Pantau pola: jika batas waktu menjadi sering, nilai apakah kueri memerlukan pengoptimalan atau jika klaster Neptunus kurang disediakan.

Pemantauan dan observabilitas

Pemantauan adalah bagian penting dari setiap strategi coba lagi. Observabilitas yang efektif membantu Anda memahami seberapa baik logika coba ulang Anda bekerja dan memberikan sinyal awal ketika sesuatu dalam beban kerja atau konfigurasi cluster Anda perlu diperhatikan.

MainRequestQueuePendingRequests

CloudWatch Metrik ini melacak jumlah permintaan yang menunggu di antrean masukan Neptunus. Nilai yang meningkat menunjukkan bahwa kueri sedang dicadangkan, yang bisa menjadi tanda

pertengkaran yang berlebihan, sumber daya yang kurang disediakan, atau badai coba lagi. Memantau metrik ini membantu Anda mengetahui kapan strategi coba ulang Anda menyebabkan atau memperparah masalah antrian, dan dapat meminta Anda untuk menyesuaikan pendekatan Anda sebelum kegagalan meningkat.

CloudWatch Metrik lainnya

Metrik Neptunus lainnya `CPUUtilization` sepertiTotalRequestsPerSecond,, dan latensi kueri memberikan konteks tambahan. Misalnya, CPU yang tinggi dan I/O dikombinasikan dengan panjang antrian yang bertambah mungkin menunjukkan bahwa klaster Anda kelebihan beban atau kueri terlalu besar atau terlalu sering. CloudWatch alarm dapat disetel pada metrik ini untuk mengingatkan Anda akan perilaku abnormal dan membantu Anda mengkorelasikan lonjakan kesalahan atau percobaan ulang dengan batasan sumber daya yang mendasarinya.

Status dan Kueri Neptunus APIs

Neptunus Status API untuk Gremlin dan APIs analognya OpenCypheruntuk dan SPARQL memberikan tampilan real-time dari kueri yang diterima dan berjalan di cluster yang berguna untuk mendiagnosis kemacetan atau memahami dampak logika coba lagi secara real time.

Dengan menggabungkan alat pemantauan ini, Anda dapat:

- Deteksi saat percobaan ulang berkontribusi pada antrian dan penurunan kinerja.
- Identifikasi kapan harus menskalakan cluster Neptunus Anda atau mengoptimalkan kueri.
- Validasi bahwa strategi coba lagi Anda menyelesaikan kegagalan sementara tanpa menutupi masalah yang lebih dalam.
- Menerima peringatan dini tentang pertentangan yang muncul atau kelelahan sumber daya.

Pemantauan dan peringatan proaktif sangat penting untuk menjaga penyebaran Neptunus yang sehat, terutama saat konkurensi dan kompleksitas aplikasi Anda tumbuh.

Mengamankan basis data Amazon Neptunus Anda

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menggambarkan hal ini sebagai keamanan dari cloud dan keamanan di cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara berkala menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku di Amazon Neptune, lihat Layanan AWS dalam Cakupan melalui Program Kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini akan membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Neptune. Topik berikut akan menunjukkan kepada Anda cara mengonfigurasi Neptune untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya Neptunus Anda.

Topik

- Peningkatan sistem operasi Amazon Neptunus
- Melindungi data di database Amazon Neptunus Anda
- Mengautentikasi database Amazon Neptunus Anda dengan AWS Identity and Access Management
- Mengaktifkan otentikasi database IAM di Amazon Neptunus
- Menghubungkan ke database Amazon Neptunus Anda menggunakan otentikasi AWS Identity and Access Management
- Mengelola akses ke database Amazon Neptunus menggunakan kebijakan IAM
- Menggunakan peran terkait layanan untuk Amazon Neptunus

- Menggunakan kredensi sementara untuk terhubung ke Amazon Neptunus
- Pencatatan dan pemantauan penggunaan dan kinerja di Amazon Neptunus
- Amazon Neptunus dan titik akhir VPC antarmuka ()AWS PrivateLink
- Pertimbangan kepatuhan untuk Amazon Neptunus
- Membangun penyebaran Amazon Neptunus yang tangguh dan toleran terhadap bencana

Peningkatan sistem operasi Amazon Neptunus

Amazon Neptunus memastikan peningkatan berkelanjutan dalam kinerja database, keamanan, dan stabilitas melalui peningkatan OS reguler. Peningkatan ini diterapkan pada Database Neptunus dan Neptunus Analytics, mengikuti proses pembaruan terstruktur. Amazon Neptunus merilis upgrade OS setidaknya sekali per bulan.

Pembaruan OS Database Neptunus yang tidak memerlukan downtime diterapkan secara otomatis selama jendela pemeliharaan. Pembaruan OS tertentu (misalnya: peningkatan versi kernel) memerlukan instance restart. Pembaruan ini bersifat opsional dan tidak memiliki tanggal yang ditetapkan untuk menerapkannya. Namun, jika Anda tidak menerapkan pembaruan ini, pembaruan tersebut pada akhirnya mungkin diperlukan dan diterapkan secara otomatis selama jendela pemeliharaan instans Anda.

Neptune Analytics - Peningkatan OS Neptunus Analytics mulus dan tidak memerlukan tindakan pelanggan. Pembaruan ini secara otomatis diterapkan di latar belakang tanpa downtime.

Untuk menjaga keamanan dan kepatuhan, kami menyarankan Anda menerapkan semua pembaruan yang disediakan oleh Amazon Neptunus secara rutin selama jendela pemeliharaan Anda. Tetap mengikuti semua pembaruan opsional dan wajib membantu menggabungkan tambalan keamanan penting dan memastikan keselarasan dengan berbagai kewajiban kepatuhan. Versi OS yang ketinggalan zaman dapat mengakibatkan ketidakpatuhan terhadap persyaratan peraturan.

Meminimalkan downtime untuk upgrade OS yang memerlukan restart

Untuk upgrade OS yang memerlukan restart, sebaiknya Anda memperbarui instance reader di cluster terlebih dahulu, kemudian instance writer untuk memaksimalkan ketersediaan cluster Anda. Kami tidak menyarankan untuk memperbarui instance pembaca dan penulis secara bersamaan, karena Anda dapat mengalami downtime yang lebih lama jika terjadi failover.

Menerapkan upgrade OS ke instans DB Neptunus

Instans DB Neptunus terkadang memerlukan pembaruan sistem operasi. Amazon Neptunus meningkatkan sistem operasi ke versi yang lebih baru untuk meningkatkan kinerja database dan postur keamanan pelanggan secara keseluruhan. Pembaruan biasanya memerlukan waktu sekitar 10 menit. Pembaruan sistem operasi tidak akan mengubah versi mesin DB atau kelas instans DB dari instans DB.

Untuk diberi tahu saat pembaruan opsional baru tersedia, Anda dapat berlangganan `RDS-EVENT-0230` dalam kategori acara penambalan keamanan. Untuk informasi tentang berlangganan acara Amazon Neptunus, lihat Berlangganan pemberitahuan acara Neptunus.

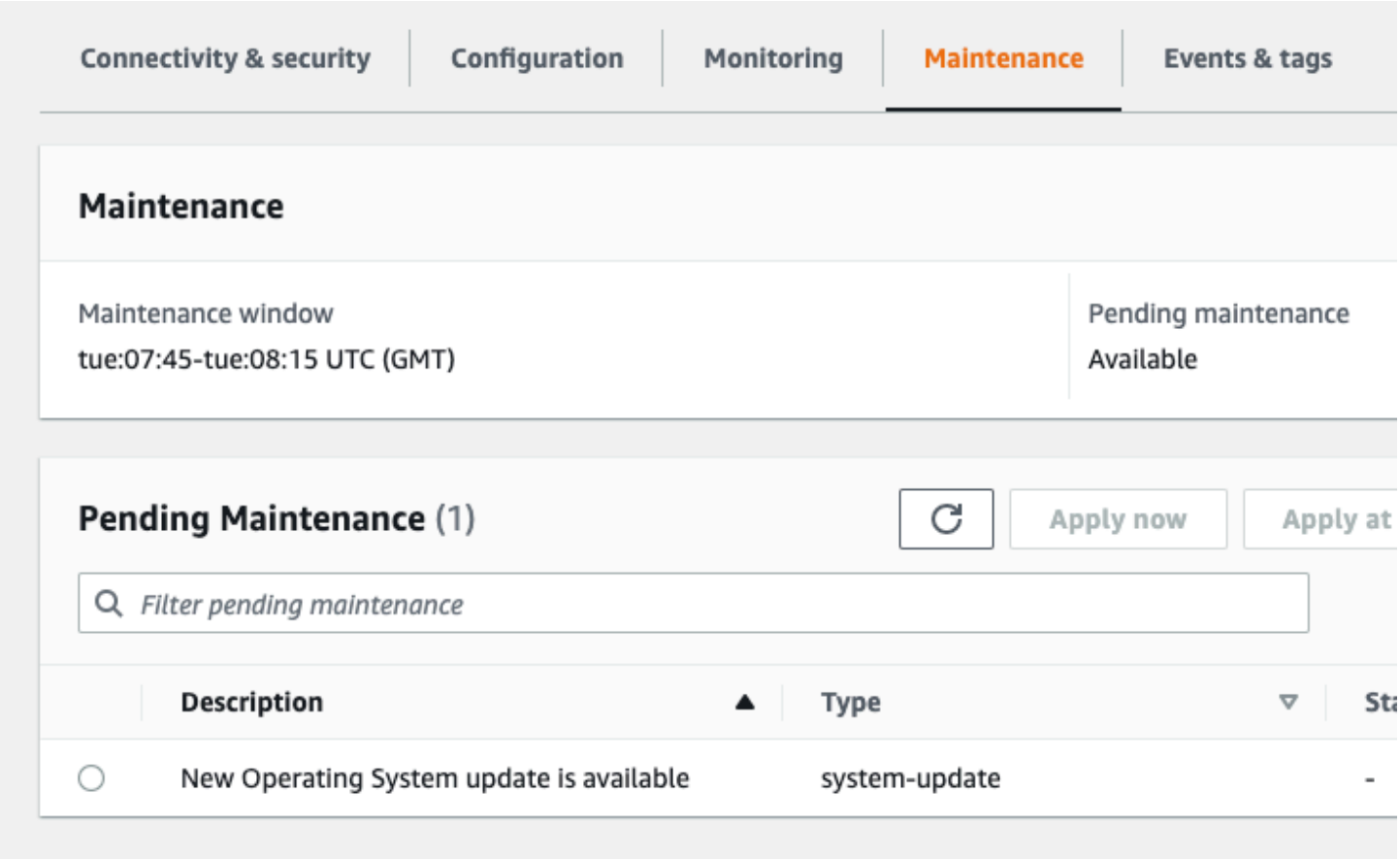
> **Important**
>
> Instans DB Amazon Neptunus Anda akan offline selama peningkatan sistem operasi. Anda dapat meminimalkan waktu henti klaster dengan memiliki klaster multi-instance. Jika Anda tidak memiliki klaster multi-instance maka Anda dapat memilih untuk membuatnya sementara dengan menambahkan instance sekunder untuk melakukan pemeliharaan ini, lalu menghapus instance pembaca tambahan setelah pemeliharaan selesai (biaya reguler untuk instance sekunder akan berlaku).

Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI untuk menentukan apakah pembaruan tersedia.

Menggunakan Konsol Manajemen AWS

Untuk menentukan apakah pembaruan tersedia menggunakan Konsol Manajemen AWS:

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Clusters, lalu pilih instance.
3. Pilih Pemeliharaan.
4. Di bagian Pending Maintenance, temukan pembaruan sistem operasi.



Anda dapat memilih pembaruan sistem operasi dan klik Terapkan sekarang atau Terapkan di jendela pemeliharaan berikutnya di bagian Pemeliharaan Tertunda. Jika nilai pemeliharaan adalah jendela berikutnya, tunda item pemeliharaan dengan memilih Defer upgrade. Anda tidak dapat menunda tindakan pemeliharaan jika sudah dimulai.

Atau, Anda dapat memilih instance dari daftar cluster dengan mengklik Cluster di panel navigasi dan pilih Terapkan sekarang atau Terapkan di jendela pemeliharaan berikutnya dari menu Tindakan.

Menggunakan AWS CLI

Untuk menentukan apakah pembaruan tersedia menggunakan AWS CLI, panggil `describe-pending-maintenance-actions` perintah:

```
aws neptune describe-pending-maintenance-actions
```

```
{
    "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:myneptune",
    "PendingMaintenanceActionDetails": [
```

```
        {
            "Action": "system-update",
            "Description": "New Operating System update is available"
        }
    ]
}
```

Untuk menerapkan Pembaruan sistem operasi, hubungi `apply-pending-maintenance-action` perintah:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --resource-identifier (ARN of your DB instance) \
    --opt-in-type immediate
```

Melindungi data di database Amazon Neptunus Anda

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di Amazon Neptunus. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensyal dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan AWS sumber daya. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat Bekerja dengan CloudTrail jejak di AWS CloudTrail Panduan Pengguna.

- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-3.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Neptunus atau Layanan AWS lainnya menggunakan konsol, API AWS CLI, atau. AWS SDKs Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

> **Important**
>
> TLS 1.3 hanya didukung untuk mesin Neptunus versi 1.3.2.0 dan yang lebih tinggi.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengelola Neptunus melalui jaringan. Klien harus mendukung Transport Layer Security (TLS) 1.2 atau lebih baru menggunakan rangkaian penyandian yang kuat, seperti yang dijelaskan dalam Mengenkripsi terhubung ke database Neptunus. Sebagian besar sistem modern seperti Java 7 dan versi yang lebih baru mendukung mode ini.

Bagian berikut menjelaskan lebih lanjut bagaimana data Neptune dilindungi.

Topik

- Mengamankan database Amazon Neptunus Anda dengan Amazon VPC
- Mengenkripsi koneksi ke database Amazon Neptunus Anda dengan SSL/HTTPS
- Mengenkripsi data saat istirahat di database Amazon Neptunus Anda

Mengamankan database Amazon Neptunus Anda dengan Amazon VPC

Cluster Amazon Neptune DB hanya dapat dibuat di Amazon Virtual Private Cloud (Amazon VPC), dan titik akhirnya dapat diakses dalam VPC itu, biasanya dari instans Amazon Elastic Compute Cloud (Amazon EC2) yang berjalan di VPC itu. Atau, dapat diakses menggunakan titik akhir publik. Untuk informasi selengkapnya tentang titik akhir publik, lihat titik akhir publik Neptunus.

Anda dapat mengamankan data Neptunus Anda dengan membatasi akses ke VPC tempat cluster DB Neptunus Anda berada, seperti yang dijelaskan dalam. Menghubungkan ke cluster Amazon Neptunus

Mengenkripsi koneksi ke database Amazon Neptunus Anda dengan SSL/HTTPS

Amazon Neptunus hanya mengizinkan koneksi Secure Sockets Layer (SSL) melalui HTTPS ke instans atau titik akhir cluster apa pun.

Neptunus membutuhkan setidaknya TLS versi 1.2, menggunakan cipher suite kuat berikut:

- `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`

Dimulai dengan Neptunus engine versi 1.3.2.0, Neptunus mendukung TLS versi 1.3 menggunakan cipher suite berikut:

- TLS_AES_128_GCM_ SHA256
- TLS_AES_256_GCM_ SHA384

Bahkan saat koneksi HTTP diperbolehkan dalam versi mesin sebelumnya, setiap klaster DB yang menggunakan grup parameter klaster DB baru diperlukan untuk menggunakan SSL secara default. Untuk melindungi data Anda, titik akhir Neptunus dalam *`1.0.4.0`* versi engine dan di atasnya hanya mendukung permintaan HTTPS. Untuk informasi selengkapnya, lihat Menggunakan titik akhir HTTP REST untuk menyambung ke instans DB Neptune.

Neptune secara otomatis menyediakan sertifikat SSL untuk instans DB Neptune Anda. Anda tidak perlu meminta sertifikat apa pun. Sertifikat disediakan saat Anda membuat instans baru.

Neptunus menetapkan sertifikat SSL wildcard tunggal ke instans di akun Anda untuk setiap Wilayah. AWS Sertifikat menyediakan entri untuk titik akhir klaster, titik akhir baca-saja klaster, dan titik akhir instans.

Detail Sertifikat

Entri berikut disertakan dalam sertifikat yang disediakan:

- Titik akhir klaster — `*.cluster-a1b2c3d4wxyz.region.neptune.amazonaws.com`
- Titik akhir hanya-baca — `*.cluster-ro-a1b2c3d4wxyz.region.neptune.amazonaws.com`
- Titik akhir instans — `*.a1b2c3d4wxyz.region.neptune.amazonaws.com`

Hanya entri yang tercantum di sini yang didukung.

Koneksi Proksi

Sertifikat hanya mendukung nama host yang tercantum di bagian sebelumnya.

Jika Anda menggunakan penyeimbang beban atau server proxy (seperti HAProxy), Anda harus menggunakan penghentian SSL dan memiliki sertifikat SSL Anda sendiri di server proxy.

Passthrough SSL tidak bekerja karena sertifikat SSL yang disediakan tidak cocok dengan nama host server proxy.

Sertifikat CA Akar

Sertifikat untuk instans Neptune biasanya divalidasi menggunakan penyimpanan kepercayaan lokal dari sistem operasi atau SDK (seperti Java SDK).

Jika Anda perlu memberikan sertifikat akar secara manual, Anda dapat men-download Sertifikat CA Akar Amazon dalam format PEM dari Amazon Trust Services Policy Repository.

Informasi Selengkapnya

Untuk informasi selengkapnya tentang menghubungkan ke titik akhir Neptune dengan SSL, lihat the section called “Menginstal konsol Gremlin” dan the section called “HTTP REST”.

Mengenkripsi data saat istirahat di database Amazon Neptunus Anda

Instans yang dienkripsi oleh Neptune memberikan lapisan perlindungan data tambahan dengan membantu mengamankan data Anda dari akses yang tidak sah ke penyimpanan yang mendasari. Anda dapat menggunakan enkripsi Neptune untuk meningkatkan perlindungan data aplikasi Anda yang disebarkan di cloud. Anda juga dapat menggunakannya untuk memenuhi persyaratan kepatuhan untuk data-at-rest enkripsi.

Untuk mengelola kunci yang digunakan untuk mengenkripsi dan mendekripsi sumber daya Neptunus Anda, Anda menggunakan ().AWS Key Management ServiceAWS KMS AWS KMS menggabungkan perangkat keras dan perangkat lunak yang aman dan sangat tersedia untuk menyediakan sistem manajemen kunci yang diskalakan untuk cloud. Dengan menggunakan AWS KMS, Anda dapat membuat kunci enkripsi dan menentukan kebijakan yang mengontrol bagaimana kunci ini dapat digunakan. AWS KMS mendukung AWS CloudTrail, sehingga Anda dapat mengaudit penggunaan kunci untuk memverifikasi bahwa kunci sedang digunakan dengan tepat. Anda dapat menggunakan AWS KMS tombol dalam kombinasi dengan Neptunus dan layanan yang AWS didukung seperti Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3), Amazon Elastic Block Store (Amazon EBS), dan Amazon Redshift. Untuk daftar layanan yang mendukung AWS KMS, lihat Cara Penggunaan AWS Layanan AWS KMS di Panduan AWS Key Management Service Pengembang.

Untuk instans yang dienkripsi Neptune, semua log, cadangan, dan tangkapan layar dienkripsi.

Mengaktifkan Enkripsi untuk instans DB Neptune

Untuk mengaktifkan enkripsi untuk instans Neptune DB baru, pilih Ya dalam bagian Aktifkan enkripsi pada konsol Neptune. Untuk informasi tentang membuat instans DB Neptune, lihat Membuat cluster Amazon Neptunus.

Saat Anda membuat instans DB Neptunus terenkripsi, Anda juga dapat menyediakan pengenal kunci untuk AWS KMS kunci enkripsi Anda. Jika Anda tidak menentukan pengenal AWS KMS kunci, Neptunus menggunakan kunci enkripsi Amazon RDS default `aws/rds` () untuk instans DB Neptunus baru Anda. AWS KMS membuat kunci enkripsi default Anda untuk Neptunus untuk akun Anda. AWS AWS Akun Anda memiliki kunci enkripsi default yang berbeda untuk setiap AWS Wilayah.

Setelah membuat instans DB Neptune terenkripsi, Anda tidak dapat mengubah kunci enkripsi untuk instans tersebut. Oleh karena itu, pastikan untuk menentukan persyaratan kunci enkripsi Anda sebelum membuat instans DB Neptune terenkripsi Anda.

Anda dapat menggunakan Amazon Resource Name (ARN) untuk kunci dari akun lain untuk mengenkripsi instans DB Neptune. Jika Anda membuat instans DB Neptunus dengan akun AWS yang sama yang memiliki kunci enkripsi yang digunakan untuk mengenkripsi instans DB Neptunus baru itu, ID kunci yang Anda lewati dapat AWS KMS menjadi alias kunci alih-alih ARN AWS KMS kunci. AWS KMS

> **Important**
>
> Jika Neptune kehilangan akses ke kunci enkripsi untuk instans DB Neptune—misalnya, ketika akses Neptune ke kunci dicabut—instans DB terenkripsi ditempatkan ke status terminal dan hanya dapat dipulihkan dari cadangan. Kami sangat menyarankan agar Anda selalu mengaktifkan pencadangan untuk instans NeptuneDB terenkripsi untuk mengamankan dari kehilangan data terenkripsi di basis data Anda.

Izin kunci yang dibutuhkan saat mengaktifkan enkripsi

Pengguna IAM atau IAM role yang menciptakan instans DB Neptune terenkripsi harus memiliki setidaknya izin berikut untuk kunci KMS:

- `"kms:Encrypt"`
- `"kms:Decrypt"`
- `"kms:GenerateDataKey"`
- `"kms:ReEncryptTo"`
- `"kms:GenerateDataKeyWithoutPlaintext"`
- `"kms:CreateGrant"`
- `"kms:ReEncryptFrom"`
- `"kms:DescribeKey"`

Berikut adalah contoh kebijakan kunci yang mencakup izin yang diperlukan:

JSON

```
{
  "Version":"2012-10-17",
  "Id": "key-consolepolicy-3",
```

```
  "Statement": [
    {
      "Sid": "Allow use of the key for Neptune",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/NeptuneFullAccess"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:ReEncryptTo",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:CreateGrant",
        "kms:ReEncryptFrom",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "rds.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "Deny use of the key for non Neptune",
      "Effect": "Deny",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/NeptuneFullAccess"
      },
      "Action": [
        "kms:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "kms:ViaService": "rds.us-east-1.amazonaws.com"
        }
      }
    }
  ]
}
```

- Pernyataan pertama memberikan akses ke semua yang diperlukan AWS KMS APIs untuk peran ini, dicakup ke Kepala Layanan RDS.
- Pernyataan kedua memperketat keamanan lebih dengan menegakkan bahwa kunci ini tidak dapat digunakan oleh peran ini untuk layanan lain AWS .

Anda juga dapat menurunkan izin `createGrant` lebih lanjut dengan menambahkan:

```
"Condition": {
  "Bool": {
    "kms:GrantIsForAWSResource": true
  }
}
```

Keterbatasan Enkripsi Neptune

Keterbatasan berikut ada untuk mengenkripsi klaster Neptune:

- Anda tidak dapat mengonversi klaster DB yang tidak terenkripsi ke klaster terenkripsi.

  Namun, Anda dapat memulihkan snapshot klaster DB yang tidak terenkripsi ke klaster DB terenkripsi. Untuk melakukannya, tentukan kunci enkripsi KMS saat Anda memulihkan dari snapshot klaster DB yang tidak terenkripsi.
- Anda tidak dapat mengonversi instans DB yang tidak terenkripsi ke klaster terenkripsi. Anda hanya dapat mengaktifkan enkripsi untuk instans DB saat Anda membuatnya.
- Selain itu, instans DB yang dienkripsi tidak dapat dimodifikasi untuk menonaktifkan enkripsi.
- Anda tidak dapat memiliki Replika Baca terenkripsi dari instans DB yang tidak dienkripsi atau Replika Baca yang tidak dienkripsi dari instans DB yang dienkripsi.
- Replika baca terenkripsi harus dienkripsikan dengan kunci yang sama dengan instans DB sumber.

Mengautentikasi database Amazon Neptunus Anda dengan AWS Identity and Access Management

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat terautentikasi (masuk) dan berwenang (memiliki izin) untuk menggunakan sumber daya Neptune. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Anda dapat menggunakan AWS Identity and Access Management (IAM) untuk mengautentikasi ke instans DB Neptunus atau cluster DB Anda. Ketika autentikasi database IAM diaktifkan, setiap permintaan harus ditandatangani menggunakan AWS Signature Version 4.

AWS Signature Version 4 menambahkan informasi otentikasi ke AWS permintaan. Untuk keamanan, semua permintaan ke klaster DB Neptune dengan autentikasi IAM diaktifkan harus ditandatangani dengan access key. Kunci ini terdiri dari access key ID dan secret access key. Autentikasi dikelola secara eksternal menggunakan kebijakan IAM.

Neptunus mengautentikasi pada koneksi, dan WebSockets untuk koneksi itu memverifikasi izin secara berkala untuk memastikan bahwa pengguna masih memiliki akses.

> **Note**
>
> - Mencabut, menghapus, atau memutar kredensials yang terkait dengan pengguna IAM tidak dianjurkan karena tidak mengakhiri koneksi yang sudah terbuka.
> - Ada batasan jumlah WebSocket koneksi bersamaan per instance database, dan berapa lama koneksi dapat tetap terbuka. Untuk informasi selengkapnya, lihat WebSockets Batas.

Penggunaan IAM Tergantung Peran Anda

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Neptunus.

Pengguna layanan — Jika Anda menggunakan layanan Neptunus untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensi dan izin yang Anda perlukan untuk menggunakan bidang data Neptunus. Karena Anda memerlukan lebih banyak akses untuk melakukan pekerjaan Anda, memahami bagaimana akses data dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda.

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya Neptunus di perusahaan Anda, Anda mungkin memiliki akses ke tindakan manajemen Neptunus, yang sesuai dengan API manajemen Neptunus. Mungkin juga tugas Anda untuk menentukan tindakan akses data Neptunus dan layanan sumber daya yang dibutuhkan pengguna untuk melakukan pekerjaan mereka. Administrator IAM kemudian dapat menerapkan kebijakan IAM untuk mengubah izin pengguna layanan Anda.

Administrator IAM — Jika Anda seorang administrator IAM, Anda perlu menulis kebijakan IAM untuk mengelola manajemen dan akses data ke Neptunus. Untuk melihat contoh kebijakan berbasis identitas Neptunus yang dapat Anda gunakan, lihat. Menggunakan berbagai jenis kebijakan IAM untuk mengontrol akses ke Neptunus

Mengautentikasi dengan Identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensyal identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensil dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), autentikasi masuk tunggal, atau kredensional. Google/Facebook Untuk informasi selengkapnya tentang cara masuk, lihat Cara masuk ke Akun AWS Anda dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS sediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat AWS Signature Version 4 untuk permintaan API dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Pengguna dan Grup IAM

Pengguna IAM adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan kredensial jangka panjang. Untuk informasi selengkapnya, lihat Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensi sementara di Panduan Pengguna IAM.

Grup IAM menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk pengguna IAM dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan beralih dari pengguna ke peran IAM (konsol) atau dengan memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat Metode untuk mengambil peran dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna terfederasi, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon EC2. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Mengaktifkan otentikasi database IAM di Amazon Neptunus

Secara default, autentikasi basis data IAM dinonaktifkan saat Anda membuat klaster DB Amazon Neptune. Anda dapat mengaktifkan autentikasi basis data IAM (atau atau menonaktifkannya kembali) menggunakan Konsol Manajemen AWS.

Untuk membuat klaster DB Neptune baru dengan autentikasi IAM menggunakan konsol, ikuti petunjuk untuk membuat klaster DB Neptune di Meluncurkan cluster DB Neptunus menggunakan Konsol Manajemen AWS.

Di halaman kedua proses pembuatan, untuk Mengaktifkan Autentikasi IAM DB, pilih Ya.

Untuk mengaktifkan atau menonaktifkan autentikasi IAM untuk instans DB atau klaster yang ada

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Pada panel navigasi, silakan pilih Klaster.
3. Pilih kluster DB Neptune yang ingin Anda ubah, dan pilih Tindakan klaster. Lalu pilih Ubah klaster.
4. Di bagian Opsi basis data, untuk Autentikasi DB IAM, pilih salah satu dari Aktifkan autorisasi DB IAM atau Tidak (untuk menonaktifkan). Lalu, pilih Lanjutkan.
5. Untuk segera menerapkan perubahan, pilih Terapkan segera.
6. Pilih Ubah klaster.

Menghubungkan ke database Amazon Neptunus Anda menggunakan otentikasi AWS Identity and Access Management

Sumber daya Amazon Neptunus yang mengaktifkan autentikasi IAM DB memerlukan semua permintaan HTTP ditandatangani menggunakan Signature Version 4. AWS Untuk informasi umum tentang penandatanganan permintaan dengan AWS Signature Versi 4, lihat Menandatangani permintaan AWS API.

AWS Signature Version 4 adalah proses untuk menambahkan informasi otentikasi ke AWS permintaan. Untuk keamanan, sebagian besar permintaan AWS harus ditandatangani dengan kunci akses, yang terdiri dari ID kunci akses dan kunci akses rahasia.

> **Note**
>
> Jika Anda menggunakan kredensial sementara, kredensial kedaluwarsa setelah interval tertentu, termasuk token sesinya.
> Anda harus memperbarui token sesi Anda ketika Anda meminta kredensial baru. Untuk informasi selengkapnya, lihat Menggunakan Kredensil Keamanan Sementara untuk Meminta Akses ke AWS Sumber Daya.

> **Important**
>
> Mengakses Neptune dengan autentikasi berbasis IAM mengharuskan Anda membuat permintaan HTTP dan menandatangani permintaan itu sendiri.

Cara Kerja Tanda Tangan Version 4

1. Anda membuat permintaan kanonik.
2. Anda menggunakan permintaan kanonik dan beberapa informasi lainnya untuk membuat. string-to-sign
3. Anda menggunakan kunci akses AWS rahasia Anda untuk mendapatkan kunci penandatanganan, dan kemudian menggunakan kunci penandatanganan itu dan string-to-sign untuk membuat tanda tangan.
4. Anda menambahkan tanda tangan yang dihasilkan untuk permintaan HTTP di header atau sebagai parameter string kueri.

Ketika Neptune menerima permintaan, ia melakukan langkah yang sama seperti yang Anda lakukan untuk menghitung tanda tangan. Neptune kemudian membandingkan tanda tangan yang dihitung dengan yang Anda kirim bersama permintaan. Jika tanda tangan cocok, permintaan diproses. Jika tanda tangan tidak cocok, permintaan ditolak.

Untuk informasi umum tentang penandatanganan permintaan dengan AWS Tanda Tangan Versi 4, lihat Proses Penandatanganan Versi Tanda Tangan 4 di Referensi Umum AWS.

Bagian berikut berisi contoh yang menunjukkan cara mengirim permintaan yang ditandatangani ke titik akhir Gremlin dan SPARQL dari instans DB Neptune dengan autentikasi IAM diaktifkan.

Topik

- Prasyarat untuk menghubungkan database Amazon Neptunus menggunakan otentikasi IAM
- Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dari baris perintah
- Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan konsol Gremlin
- Menghubungkan ke database Amazon Neptunus menggunakan IAM dengan Gremlin Java
- Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Java dan SPARQL
- Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan SPARQL dan Node.js
- Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Python
- Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Python Gremlin
- Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Gremlin JavaScript
- Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Gremlin Go
- Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Gremlin.NET

Prasyarat untuk menghubungkan database Amazon Neptunus menggunakan otentikasi IAM

Berikut ini adalah petunjuk untuk menginstal Apache Maven dan Java 8 pada instans Amazon EC2. Ini diperlukan untuk sampel autentikasi Amazon Neptune Signature Versi 4.

Untuk menginstal Apache Maven dan Java 8 pada instans EC2 Anda

1. Hubungkan ke instans Amazon EC2 dengan klien SSH.
2. Install Apache Maven di instans EC2 Anda. Jika menggunakan Amazon Linux 2023 (lebih disukai), gunakan:

   ```
   sudo dnf update -y
   sudo dnf install maven -y
   ```

   Jika menggunakan Amazon Linux 2, unduh biner terbaru dari https://maven.apache.org/download.cgi:

   ```
   sudo yum remove maven -y
   wget https://dlcdn.apache.org/maven/maven-3/ <version>/binaries/apache-maven-
   <version>-bin.tar.gz
   sudo tar -xzf apache-maven-<version>-bin.tar.gz -C /opt/
   sudo ln -sf /opt/apache-maven-<version> /opt/maven
   echo 'export MAVEN_HOME=/opt/maven' >> ~/.bashrc
   echo 'export PATH=$MAVEN_HOME/bin:$PATH' >> ~/.bashrc
   source ~/.bashrc
   ```
3. Pustaka Gremlin membutuhkan Java 8. Masukkan hal berikut ini untuk menginstal Java 8 pada instans EC2 Anda.

   ```
   sudo yum install java-1.8.0-devel
   ```
4. Masukkan hal berikut untuk mengatur Java 8 sebagai runtime default pada instans EC2 Anda.

   ```
   sudo /usr/sbin/alternatives --config java
   ```

   Saat diminta, masukkan nomor untuk Java 8.
5. Masukkan hal berikut untuk mengatur Java 8 sebagai compiler default pada instans EC2 Anda.

   ```
   sudo /usr/sbin/alternatives --config javac
   ```

   Saat diminta, masukkan nomor untuk Java 8.

Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dari baris perintah

Memiliki alat baris perintah untuk mengirimkan kueri ke cluster DB Neptunus Anda sangat berguna, seperti yang diilustrasikan dalam banyak contoh dalam dokumentasi ini. Alat curl adalah opsi yang sangat baik untuk berkomunikasi dengan titik akhir Neptunus jika otentikasi IAM tidak diaktifkan.

Namun, untuk menjaga keamanan data Anda, yang terbaik adalah mengaktifkan otentikasi IAM.

Ketika autentikasi IAM diaktifkan, setiap permintaan harus ditandatangani menggunakan Signature Version 4 (Sig4). Alat baris perintah awscurl pihak ketiga menggunakan sintaks yang sama seperti, dan dapat menandatangani kueri menggunakan `curl` penandatanganan Sig4. Menggunakan `awscurl`Bagian di bawah ini menjelaskan cara menggunakan `awscurl` secara aman dengan kredensil sementara.

Menyiapkan alat baris perintah untuk menggunakan HTTPS

Neptunus mengharuskan semua koneksi menggunakan HTTPS. Alat baris perintah apa pun seperti `curl` atau `awscurl` memerlukan akses ke sertifikat yang sesuai untuk menggunakan HTTPS. Selama `curl` atau `awscurl` dapat menemukan sertifikat yang sesuai, mereka menangani koneksi HTTPS seperti koneksi HTTP, tanpa memerlukan parameter tambahan. Contoh dalam dokumentasi ini didasarkan pada skenario tersebut.

Untuk mempelajari cara mendapatkan sertifikat tersebut dan cara memformatnya dengan benar ke dalam penyimpanan sertifikat otoritas sertifikat (CA) yang `curl` dapat digunakan, lihat Verifikasi Sertifikat SSL dalam `curl` dokumentasi.

Anda kemudian dapat menentukan lokasi penyimpanan sertifikat CA ini menggunakan variabel lingkungan `CURL_CA_BUNDLE`. Pada Windows, `curl` secara otomatis mencarinya dalam sebuah file bernama `curl-ca-bundle.crt`. Ia pertama mencari dalam direktori yang sama dengan `curl.exe` dan kemudian di tempat lain di jalurnya. Untuk informasi lebih lanjut, lihat Verifikasi Sertifikat SSL.

Menggunakan **`awscurl`** dengan kredensyal sementara untuk terhubung dengan aman ke cluster DB dengan otentikasi IAM diaktifkan

Alat awscurl menggunakan sintaks yang sama seperti`curl`, tetapi membutuhkan informasi tambahan juga:

- **`--access_key`**— Kunci akses yang valid. Jika tidak disediakan menggunakan parameter ini, itu harus disediakan dalam variabel `AWS_ACCESS_KEY_ID` lingkungan, atau dalam file konfigurasi.

- **`--secret_key`**— Kunci rahasia yang valid sesuai dengan kunci akses. Jika tidak disediakan menggunakan parameter ini, itu harus disediakan dalam variabel `AWS_SECRET_ACCESS_KEY` lingkungan, atau dalam file konfigurasi.
- **`--security_token`**— Token sesi yang valid. Jika tidak disediakan menggunakan parameter ini, itu harus disediakan dalam variabel `AWS_SECURITY_TOKEN` lingkungan, atau dalam file konfigurasi.

Di masa lalu, itu adalah praktik umum untuk menggunakan kredensil persisten dengan`awscurl`, seperti kredensil pengguna IAM atau bahkan kredensil root, tetapi ini tidak disarankan. Sebagai gantinya, buat kredensyal sementara menggunakan salah satu AWS Security Token Service (STS) APIs, atau salah satu pembungkusnya AWS CLI .

Yang terbaik adalah menempatkan`AccessKeyId`,`SecretAccessKey`, dan `SessionToken` nilai yang dikembalikan oleh panggilan STS ke variabel lingkungan yang sesuai di sesi shell Anda daripada ke dalam file konfigurasi. Kemudian, ketika shell berakhir, kredensyal secara otomatis dibuang, yang tidak terjadi dengan file konfigurasi. Demikian pula, jangan meminta durasi yang lebih lama untuk kredensyal sementara daripada yang mungkin Anda butuhkan.

Contoh berikut menunjukkan langkah-langkah yang mungkin Anda ambil dalam shell Linux untuk mendapatkan kredensyal sementara yang baik selama setengah jam menggunakan sts assume-role, dan kemudian menempatkannya dalam variabel lingkungan di mana dapat menemukannya: `awscurl`

```
aws sts assume-role \
    --duration-seconds 1800 \
    --role-arn "arn:aws:iam::(account-id):role/(rolename)" \
    --role-session-name AWSCLI-Session > $output
AccessKeyId=$(echo $output | jq '.Credentials''.AccessKeyId')
SecretAccessKey=$(echo $output | jq '.Credentials''.SecretAccessKey')
SessionToken=$(echo $output | jq '.Credentials''.SessionToken')

export AWS_ACCESS_KEY_ID=$AccessKeyId
export AWS_SECRET_ACCESS_KEY=$SecretAccessKey
export AWS_SESSION_TOKEN=$SessionToken
```

Anda kemudian dapat menggunakan `awscurl` untuk membuat permintaan yang ditandatangani ke cluster DB Anda seperti ini:

```
awscurl (your cluster endpoint):8182/status \
```

```
    --region us-east-1 \
    --service neptune-db
```

Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan konsol Gremlin

Untuk menyambung ke Amazon Neptunus menggunakan konsol Gremlin dengan otentikasi Signature Version 4, Anda `requestInterceptor()` gunakan untuk menyambungkan tanda tangan SiGv4 ke sambungan yang dibuat oleh perintah. `:remote` Ini mengharuskan Anda untuk mengkonfigurasi `Cluster` objek secara manual dan kemudian meneruskannya ke `:remote` perintah.

Perhatikan bahwa ini sangat berbeda dari situasi khas di mana `:remote` perintah mengambil file konfigurasi untuk membentuk koneksi. Pendekatan file konfigurasi tidak akan berfungsi karena `requestInterceptor()` harus diatur secara terprogram, dan tidak dapat memuat konfigurasinya dari file.

> **Note**
>
> Contoh berikut digunakan`requestInterceptor()`, yang diperkenalkan di TinkerPop 3.6.6. Jika Anda menggunakan TinkerPop versi lebih awal dari 3.6.6 (tetapi 3.5.5 atau lebih tinggi), gunakan `handshakeInterceptor()` alih-alih `requestInterceptor()` dalam contoh kode di bawah ini.

Prasyarat berikut diperlukan:

- Anda harus memiliki kredensyal IAM yang diperlukan untuk menandatangani permintaan. Lihat Menggunakan rantai penyedia kredensi default di Panduan AWS SDK untuk Java Pengembang.
- Anda harus menginstal versi konsol Gremlin yang kompatibel dengan versi mesin Neptunus yang digunakan oleh cluster DB Anda.

Jika Anda menggunakan kredensil sementara, kredensil tersebut akan kedaluwarsa setelah interval tertentu, seperti halnya token sesi, jadi Anda harus memperbarui token sesi saat Anda meminta kredensil baru. Lihat Menggunakan kredensil keamanan sementara untuk meminta akses ke AWS sumber daya di Panduan Pengguna IAM.

Untuk bantuan menghubungkan menggunakan SSL/TLS, lihat. Konfigurasi SSL/TLS

Hubungkan konsol Gremlin dengan penandatanganan Sig4

1. Mulai konsol Gremlin:

```
$ bin/gremlin.sh
```

2. Pada `gremlin>` prompt, instal `amazon-neptune-sigv4-signer` perpustakaan (ini hanya perlu dilakukan sekali untuk konsol):

```
:install com.amazonaws amazon-neptune-sigv4-signer 2.4.0
```

   Jika Anda mengalami masalah dengan langkah ini, mungkin membantu untuk berkonsultasi dengan TinkerPop dokumentasi tentang konfigurasi Grape.

   > **Note**
   >
   > Jika Anda menggunakan proxy HTTP, Anda mungkin mengalami kesalahan dengan langkah ini di mana `:install` perintah tidak selesai. Untuk mengatasi masalah ini, jalankan perintah berikut untuk memberi tahu konsol tentang proxy:
   >
   > ```
   > System.setProperty("https.proxyHost", "(the proxy IP address)")
   > System.setProperty("https.proxyPort", "(the proxy port)")
   > ```

3. Impor kelas yang diperlukan untuk menangani masuk `kerequestInterceptor()`:

```
:import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider
:import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer
```

4. Jika Anda menggunakan kredensi sementara, Anda juga perlu menyediakan Token Sesi Anda sebagai berikut:

```
System.setProperty("aws.sessionToken","(your session token)")
```

5. Jika Anda belum menetapkan kredensi akun Anda, Anda dapat menetapkannya sebagai berikut:

```
System.setProperty("aws.accessKeyId","(your access key)")
System.setProperty("aws.secretKey","(your secret key)")
```

6. Buat `Cluster` objek secara manual untuk terhubung ke Neptunus:

```
cluster = Cluster.build("(host name)")  \
                .enableSsl(true) \
                .requestInterceptor { r ->  \
                  def sigV4Signer = new NeptuneNettyHttpSigV4Signer("(Amazon
 region)", \
                                        DefaultCredentialsProvider.create()); \
                  sigV4Signer.signRequest(r); \
                  return r; } \
                .create()
```

Untuk bantuan menemukan nama host dari instans DB Neptunus, lihat. Menghubungkan ke Titik Akhir Amazon Neptune.

7. Buat `:remote` koneksi menggunakan nama variabel `Cluster` objek pada langkah sebelumnya:

```
:remote connect tinkerpop.server cluster
```

8. Masukkan perintah berikut untuk beralih ke mode jarak jauh. Ini mengirimkan semua kueri Gremlin ke koneksi jarak jauh:

```
:remote console
```

Menghubungkan ke database Amazon Neptunus menggunakan IAM dengan Gremlin Java

Berikut adalah contoh cara terhubung ke Neptunus menggunakan Gremlin Java API dengan penandatanganan Sig4 (ini mengasumsikan pengetahuan umum tentang penggunaan Maven). Contoh ini menggunakan library Amazon Neptune SiGv4 Signer untuk membantu penandatanganan permintaan. Pertama, tentukan dependensi sebagai bagian dari file: `pom.xml`

> **Note**
>
> Contoh berikut digunakan`requestInterceptor()`, yang diperkenalkan di TinkerPop 3.6.6. Jika Anda menggunakan TinkerPop versi lebih awal dari 3.6.6 (tetapi 3.5.5 atau lebih tinggi), gunakan `handshakeInterceptor()` alih-alih `requestInterceptor()` dalam contoh kode di bawah ini.

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>amazon-neptune-sigv4-signer</artifactId>
  <version>3.1.0</version>
</dependency>
```

Amazon Neptune SiGv4 Signer mendukung kedua versi 1.x dan 2.x dari Java SDK. AWS Contoh berikut menggunakan 2.x, di mana `DefaultCredentialsProvider` adalah sebuah `software.amazon.awssdk.auth.credentials.AwsCredentialsProvider` instance. Jika Anda memutakhirkan dari 1.x ke 2.x, lihat perubahan penyedia Credentials dalam dokumentasi SDK for AWS Java 2.x.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer;
import com.amazonaws.neptune.auth.NeptuneSigV4SignerException;

 ...

System.setProperty("aws.accessKeyId","your-access-key");
System.setProperty("aws.secretKey","your-secret-key");

 ...

Cluster cluster = Cluster.build((your cluster))
                 .enableSsl(true)
                 .requestInterceptor( r ->
                  {
                    try {
                      NeptuneNettyHttpSigV4Signer sigV4Signer =
                        new NeptuneNettyHttpSigV4Signer("(your region)",
 DefaultCredentialsProvider.create());
                      sigV4Signer.signRequest(r);
                    } catch (NeptuneSigV4SignerException e) {
                      throw new RuntimeException("Exception occurred while signing the
 request", e);
                    }
                    return r;
                  }
                 ).create();
try {
  Client client = cluster.connect();
  client.submit("g.V().has('code','IAD')").all().get();
```

```
} catch (Exception e) {
  throw new RuntimeException("Exception occurred while connecting to cluster", e);
}
```

Autentikasi IAM lintas akun

Amazon Neptunus mendukung otentikasi IAM lintas akun melalui penggunaan asumsi peran, juga kadang-kadang disebut sebagai rantai peran. Untuk menyediakan akses ke cluster Neptunus dari aplikasi yang dihosting di akun yang berbeda: AWS

- Buat pengguna atau peran IAM baru di AWS akun aplikasi, dengan kebijakan kepercayaan yang memungkinkan pengguna atau peran untuk mengambil peran IAM lain. Tetapkan peran ini ke komputasi yang menghosting aplikasi (instans EC2, fungsi Lambda, Tugas ECS, dll.).

  JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "assumeRolePolicy",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::111122223333:role/role-name"
    }
  ]
}
```

- Buat peran IAM baru di akun AWS database Neptunus yang memungkinkan akses ke database Neptunus dan memungkinkan asumsi peran dari pengguna/peran IAM akun aplikasi. Gunakan kebijakan kepercayaan:

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": [
                    "(ARN of application account IAM user or role)"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {}
        }
    ]
}
```

- Gunakan contoh kode berikut sebagai panduan tentang cara menggunakan dua peran ini untuk memungkinkan aplikasi mengakses Neptunus. Dalam contoh ini, peran akun aplikasi akan diasumsikan melalui DefaultCredentialProviderChainsaat membuat`STSclient`. Kemudian `STSclient` digunakan melalui `STSAssumeRoleSessionCredentialsProvider` untuk mengambil peran yang dihosting di akun database AWS Neptunus.

```
public static void main( String[] args )
  {

    /*
     * Establish an STS client from the application account.
     */
    AWSSecurityTokenService client = AWSSecurityTokenServiceClientBuilder
        .standard()
        .build();

    /*
     * Define the role ARN that you will be assuming in the database account where
 the Neptune cluster resides.
     */
    String roleArnToAssume = "arn:aws:iam::012345678901:role/
CrossAccountNeptuneRole";
    String crossAccountSessionName = "cross-account-session-" + UUID.randomUUID();

    /*
     * Change the Credentials Provider in the SigV4 Signer to use the STSAssumeRole
 Provider and provide it
     * with both the role to be assumed, the original STS client, and a session name
 (which can be
     * arbitrary.)
     */
    Cluster cluster = Cluster.build()
                .addContactPoint("neptune-cluster.us-west-2.neptune.amazonaws.com")
```

```
                .enableSsl(true)
                .port(8182)
                .requestInterceptor( r ->
                 {
                   try {
                     NeptuneNettyHttpSigV4Signer sigV4Signer =
                       // new NeptuneNettyHttpSigV4Signer("us-west-2",
DefaultCredentialsProvider.create());
                       new NeptuneNettyHttpSigV4Signer(
                               "us-west-2",
                                new STSAssumeRoleSessionCredentialsProvider
                                   .Builder(roleArnToAssume,
crossAccountSessionName)
                                       .withStsClient(client)
                                       .build());
                     sigV4Signer.signRequest(r);
                   } catch (NeptuneSigV4SignerException e) {
                     throw new RuntimeException("Exception occurred while signing
the request", e);
                   }
                   return r;
                 }
                ).create();

   GraphTraversalSource g =
traversal().withRemote(DriverRemoteConnection.using(cluster));

   /* whatever application code is necessary */

   cluster.close();
 }
```

Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Java dan SPARQL

Bagian ini menunjukkan cara terhubung ke Neptunus menggunakan J RDF4 atau Apache Jena dengan otentikasi Signature Version 4.

Prasyarat

- Java 8 atau lebih tinggi.

- Apache Maven 3.3 atau lebih tinggi.

  Untuk informasi tentang menginstal prasyarat ini di instans EC2 yang menjalankan Amazon Linux, lihat Prasyarat untuk menghubungkan database Amazon Neptunus menggunakan otentikasi IAM.

- Kredensial IAM untuk menandatangani permintaan. Untuk informasi selengkapnya, lihat Menggunakan Rantai Penyedia Kredential Default dalam Panduan Developer AWS SDK untuk Java .

  > **Note**
  >
  > Jika Anda menggunakan kredensial sementara, kredensial kedaluwarsa setelah interval tertentu, termasuk token sesinya.
  > Anda harus memperbarui token sesi Anda ketika Anda meminta kredensial baru. Untuk informasi selengkapnya, lihat Menggunakan Kredensyal Keamanan Sementara untuk Meminta Akses ke AWS Sumber Daya di Panduan Pengguna IAM.

- Atur variabel `SERVICE_REGION` ke salah satu dari berikut ini, menunjukkan Wilayah instans DB Neptune Anda:
  - US East (N. Virginia): `us-east-1`
  - AS Timur (Ohio): `us-east-2`
  - US West (N. California): `us-west-1`
  - US West (Oregon): `us-west-2`
  - Canada (Central): `ca-central-1`
  - Kanada Barat (Calgary): `ca-west-1`
  - South America (São Paulo): `sa-east-1`
  - Eropa (Stockholm): `eu-north-1`
  - Eropa (Spanyol): `eu-south-2`
  - Eropa (Irlandia): `eu-west-1`
  - Eropa (London): `eu-west-2`
  - Eropa (Paris): `eu-west-3`
  - Eropa (Frankfurt): `eu-central-1`
  - Timur Tengah (Bahrain): `me-south-1`
  - Timur Tengah (UEA): `me-central-1`
  - Israel (Tel Aviv): `il-central-1`

- Afrika (Cape Town): `af-south-1`
- Asia Pasifik (Hong Kong): `ap-east-1`
- Asia Pacific (Tokyo): `ap-northeast-1`
- Asia Pasifik (Seoul): `ap-northeast-2`
- Asia Pasifik (Osaka): `ap-northeast-3`
- Asia Pacific (Singapore): `ap-southeast-1`
- Asia Pacific (Sydney): `ap-southeast-2`
- Asia Pasifik (Jakarta): `ap-southeast-3`
- Asia Pasifik (Melbourne): `ap-southeast-4`
- Asia Pasifik (Malaysia): `ap-southeast-5`
- Asia Pasifik (Mumbai): `ap-south-1`
- Asia Pasifik (Hyderabad): `ap-south-2`
- Tiongkok (Beijing): `cn-north-1`
- Tiongkok (Ningxia): `cn-northwest-1`
- AWS GovCloud (AS-Barat): `us-gov-west-1`
- AWS GovCloud (AS-Timur): `us-gov-east-1`

Untuk terhubung ke Neptunus menggunakan J RDF4 atau Apache Jena dengan penandatanganan Signature Version 4

1. Kloning repositori sampel dari. GitHub

```
git clone https://github.com/aws/amazon-neptune-sparql-java-sigv4.git
```

2. Ubah ke dalam direktori kloning.

```
cd amazon-neptune-sparql-java-sigv4
```

3. Dapatkan versi terbaru dari proyek dengan memeriksa cabang dengan tanda terbaru.

```
git checkout $(git describe --tags `git rev-list --tags --max-count=1`)
```

4. Masukkan salah satu perintah berikut untuk mengompilasi dan menjalankan kode contoh.

Ganti *your-neptune-endpoint* dengan nama host atau alamat IP instans DB Neptunus Anda. Port default adalah 8182.

> **Note**
>
> Untuk informasi tentang menemukan nama host instans DB Neptune Anda, lihat bagian Menghubungkan ke Titik Akhir Amazon Neptune..

Gerhana J RDF4

Masukkan yang berikut ini untuk menjalankan contoh RDF4 J.

```
mvn compile exec:java \
    -Dexec.mainClass="com.amazonaws.neptune.client.rdf4j.NeptuneRdf4JSigV4Example"
 \
    -Dexec.args="https://your-neptune-endpoint:portsparql region-name"
```

Apache Jena

Masukkan berikut ini untuk menjalankan contoh Apache Jena.

```
mvn compile exec:java \
    -Dexec.mainClass="com.amazonaws.neptune.client.jena.NeptuneJenaSigV4Example" \
    -Dexec.args="https://your-neptune-endpoint:port"
```

5. Untuk melihat kode sumber untuk contoh, lihat contoh dalam direktori `src/main/java/com/amazonaws/neptune/client/`.

Untuk menggunakan driver penandatanganan SigV4 di aplikasi Java Anda sendiri, tambahkan paket Maven `amazon-neptune-sigv4-signer` ke bagian `<dependencies>` dari `pom.xml` Anda. Sebaiknya Anda menggunakan contoh sebagai titik awal.

Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan SPARQL dan Node.js

Kueri menggunakan penandatanganan Signature V4 dan AWS SDK untuk Javascript V3

Berikut adalah contoh cara menghubungkan ke Neptunus SPARQL menggunakan Node.js dengan otentikasi Signature Version 4 dan SDK untuk Javascript V3: AWS

```
const { HttpRequest }  = require('@smithy/protocol-http');
const { fromNodeProviderChain } = require('@aws-sdk/credential-providers');
const { SignatureV4 } = require('@smithy/signature-v4');
const { Sha256 } = require('@aws-crypto/sha256-universal');
const https = require('https');

var region = 'us-west-2'; // e.g. us-west-1
var neptune_endpoint = 'your-Neptune-cluster-endpoint';  // like: 'cluster-
id.region.neptune.amazonaws.com'
var query = `query=PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX class: <http://aws.amazon.com/neptune/csv2rdf/class/>
PREFIX resource: <http://aws.amazon.com/neptune/csv2rdf/resource/>
PREFIX prop: <http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/>
PREFIX objprop: <http://aws.amazon.com/neptune/csv2rdf/objectProperty/>

SELECT ?movies ?title WHERE {
  ?jel prop:name "James Earl Jones" .
  ?movies ?p2 ?jel .
  ?movies prop:title ?title
} LIMIT 10`;

runQuery(query);

function runQuery(q) {
  var request = new HttpRequest({
    hostname: neptune_endpoint,
    port: 8182,
    path: 'sparql',
    body: encodeURI(query),
    headers: {
      'Content-Type': 'application/x-www-form-urlencoded',
      'host': neptune_endpoint + ':8182',
    },
```

```
    method: 'POST',
  });

  const credentialProvider = fromNodeProviderChain();
  let credentials = credentialProvider();
  credentials.then(
    (cred)=>{
      var signer = new SignatureV4({credentials: cred, region: region, sha256: Sha256,
 service: 'neptune-db'});
      signer.sign(request).then(
        (req)=>{
          var responseBody = '';
          var sendreq = https.request(
            {
              host: req.hostname,
              port: req.port,
              path: req.path,
              method: req.method,
              headers: req.headers,
            },
          (res) => {
            res.on('data', (chunk) => { responseBody += chunk; });
            res.on('end', () => {
                console.log(JSON.parse(responseBody));
            });
          });
          sendreq.write(req.body);
          sendreq.end();
        }
      );
    },
    (err)=>{
      console.error(err);
    }
  );
}
```

Kueri menggunakan penandatanganan Signature V4 dan AWS SDK untuk Javascript V2

Berikut adalah contoh cara menghubungkan ke Neptunus SPARQL menggunakan Node.js dengan otentikasi Signature Version 4 dan SDK untuk Javascript V2: AWS

```
var AWS = require('aws-sdk');

var region = 'us-west-2'; // e.g. us-west-1
var neptune_endpoint = 'your-Neptune-cluster-endpoint';  // like: 'cluster-
id.region.neptune.amazonaws.com'
var query = `query=PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX class: <http://aws.amazon.com/neptune/csv2rdf/class/>
PREFIX resource: <http://aws.amazon.com/neptune/csv2rdf/resource/>
PREFIX prop: <http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/>
PREFIX objprop: <http://aws.amazon.com/neptune/csv2rdf/objectProperty/>

SELECT ?movies ?title WHERE {
    ?jel prop:name "James Earl Jones" .
    ?movies ?p2 ?jel .
    ?movies prop:title ?title
} LIMIT 10`;

runQuery(query);

function runQuery(q) {

    var endpoint = new AWS.Endpoint(neptune_endpoint);
    endpoint.port = 8182;
    var request = new AWS.HttpRequest(endpoint, region);
    request.path += 'sparql';
    request.body = encodeURI(query);
    request.headers['Content-Type'] = 'application/x-www-form-urlencoded';
    request.headers['host'] = neptune_endpoint;
    request.method = 'POST';

    var credentials = new AWS.CredentialProviderChain();
    credentials.resolve((err, cred)=>{
        var signer = new AWS.Signers.V4(request, 'neptune-db');
        signer.addAuthorization(cred, new Date());
    });

    var client = new AWS.HttpClient();
    client.handleRequest(request, null, function(response) {
        console.log(response.statusCode + ' ' + response.statusMessage);
        var responseBody = '';
        response.on('data', function (chunk) {
            responseBody += chunk;
        });
```

```
        response.on('end', function (chunk) {
            console.log('Response body: ' + responseBody);
        });
    }, function(error) {
        console.log('Error: ' + error);
    });
}
```

Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Python

Bagian ini menunjukkan contoh program yang ditulis dengan Python yang menggambarkan bagaimana bekerja dengan Signature Versi 4 untuk Amazon Neptune. Contoh ini didasarkan pada contoh di bagian Proses Penandatanganan Versi Tanda Tangan 4 di bagian Referensi Umum Amazon Web.

Untuk bekerja dengan program contoh ini, Anda memerlukan yang berikut ini:

- Python 3.x diinstal pada komputer Anda, yang bisa didapatkan dari Situs Python. Program-program ini diuji menggunakan Python 3.6.
- Pustakan permintaan Python, yang digunakan dalam skrip contoh untuk membuat permintaan web. Cara mudah untuk menginstal paket Python adalah dengan menggunakan `pip`, yang mendapat paket dari situs indeks paket Python. Anda kemudian dapat menginstal `requests` dengan menjalankan `pip install requests` pada baris perintah.
- Kunci akses (access key ID dan secret access key) di variabel lingkungan bernama `AWS_ACCESS_KEY_ID` dan `AWS_SECRET_ACCESS_KEY`. Sebagai praktik terbaik, kami menyarankan Anda untuk tidak menanamkan kredensial dalam kode. Untuk informasi selengkapnya, lihat Praktik Terbaik untuk akun AWS di Panduan Referensi AWS Account Management .

  Wilayah klaster DB Neptune Anda dalam variabel lingkungan bernama `SERVICE_REGION`.

  Jika Anda menggunakan kredensial sementara, Anda harus menentukan `AWS_SESSION_TOKEN` selain `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, dan `SERVICE_REGION`.

  > **Note**
  >
  > Jika Anda menggunakan kredensial sementara, kredensial kedaluwarsa setelah interval tertentu, termasuk token sesinya.

Anda harus memperbarui token sesi Anda ketika Anda meminta kredensial baru. Untuk informasi selengkapnya, lihat Menggunakan Kredensial Keamanan Sementara untuk Meminta Akses ke Sumber Daya AWS.

Contoh berikut menunjukkan cara membuat permintaan yang ditandatangani ke Neptune menggunakan Python. Permintaan tersebut membuat permintaan GET atau POST. Informasi autentikasi dilewatkan menggunakan header permintaan `Authorization`.

Contoh ini juga berfungsi sebagai AWS Lambda fungsi. Untuk informasi selengkapnya, lihat the section called “Menyiapkan Lambda”.

Untuk membuat permintaan yang ditandatangani ke titik akhir Gremlin dan SPARQL Neptune

1. Buat file baru bernama `neptunesigv4.py`, lalu buka file tersebut dalam editor teks.
2. Salin kode berikut dan tempelkan ke file `neptunesigv4.py`.

```
# Amazon Neptune version 4 signing example (version v3)

# The following script requires python 3.6+
#    (sudo yum install python36 python36-virtualenv python36-pip)
# => the reason is that we're using urllib.parse() to manually encode URL
#    parameters: the problem here is that SIGV4 encoding requires whitespaces
#    to be encoded as %20 rather than not or using '+', as done by previous/
#    default versions of the library.


# See: https://docs.aws.amazon.com/general/latest/gr/sigv4_signing.html
import sys, datetime, hashlib, hmac
import requests  # pip3 install requests
import urllib
import os
import json
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import ReadOnlyCredentials
from types import SimpleNamespace
from argparse import RawTextHelpFormatter
from argparse import ArgumentParser

# Configuration. https is required.
```

```
protocol = 'https'

# The following lines enable debugging at httplib level (requests->urllib3-
>http.client)
# You will see the REQUEST, including HEADERS and DATA, and RESPONSE with HEADERS
 but without DATA.
#
# The only thing missing will be the response.body which is not logged.
#
# import logging
# from http.client import HTTPConnection
# HTTPConnection.debuglevel = 1
# logging.basicConfig()
# logging.getLogger().setLevel(logging.DEBUG)
# requests_log = logging.getLogger("requests.packages.urllib3")
# requests_log.setLevel(logging.DEBUG)
# requests_log.propagate = True


# Read AWS access key from env. variables. Best practice is NOT
# to embed credentials in code.
access_key = os.getenv('AWS_ACCESS_KEY_ID', '')
secret_key = os.getenv('AWS_SECRET_ACCESS_KEY', '')
region = os.getenv('SERVICE_REGION', '')

# AWS_SESSION_TOKEN is optional environment variable. Specify a session token only
 if you are using temporary
# security credentials.
session_token = os.getenv('AWS_SESSION_TOKEN', '')

### Note same script can be used for AWS Lambda (runtime = python3.6).
## Steps to use this python script for AWS Lambda
# 1. AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY and AWS_SESSION_TOKEN and AWS_REGION
 variables are already part of Lambda's Execution environment
#    No need to set them up explicitly.
# 3. Create Lambda deployment package https://docs.aws.amazon.com/lambda/latest/dg/
lambda-python-how-to-create-deployment-package.html
# 4. Create a Lambda function in the same VPC and assign an IAM role with neptune
 access

def lambda_handler(event, context):
    # sample_test_input = {
    #     "host": "END_POINT:8182",
    #     "method": "GET",
```

```
    #     "query_type": "gremlin",
    #     "query": "g.V().count()"
    # }

    # Lambda uses AWS_REGION instead of SERVICE_REGION
    global region
    region = os.getenv('AWS_REGION', '')

    host = event['host']
    method = event['method']
    query_type = event['query_type']
    query =  event['query']

    return make_signed_request(host, method, query_type, query)

def validate_input(method, query_type):
    # Supporting GET and POST for now:
    if (method != 'GET' and method != 'POST'):
        print('First parameter must be "GET" or "POST", but is "' + method + '".')
        sys.exit()

    # SPARQL UPDATE requires POST
    if (method == 'GET' and query_type == 'sparqlupdate'):
        print('SPARQL UPDATE is not supported in GET mode. Please choose POST.')
        sys.exit()

def get_canonical_uri_and_payload(query_type, query, method):
    # Set the stack and payload depending on query_type.
    if (query_type == 'sparql'):
        canonical_uri = '/sparql/'
        payload = {'query': query}

    elif (query_type == 'sparqlupdate'):
        canonical_uri = '/sparql/'
        payload = {'update': query}

    elif (query_type == 'gremlin'):
        canonical_uri = '/gremlin/'
        payload = {'gremlin': query}
        if (method == 'POST'):
            payload = json.dumps(payload)

    elif (query_type == 'openCypher'):
        canonical_uri = '/openCypher/'
```

```
        payload = {'query': query}

    elif (query_type == "loader"):
        canonical_uri = "/loader/"
        payload = query

    elif (query_type == "status"):
        canonical_uri = "/status/"
        payload = {}

    elif (query_type == "gremlin/status"):
        canonical_uri = "/gremlin/status/"
        payload = {}

    elif (query_type == "openCypher/status"):
        canonical_uri = "/openCypher/status/"
        payload = {}

    elif (query_type == "sparql/status"):
        canonical_uri = "/sparql/status/"
        payload = {}

    else:
        print(
            'Third parameter should be from ["gremlin", "sparql", "sparqlupdate",
 "loader", "status] but is "' + query_type + '".')
        sys.exit()
    ## return output as tuple
    return canonical_uri, payload

def make_signed_request(host, method, query_type, query):
    service = 'neptune-db'
    endpoint = protocol + '://' + host

    print()
    print('+++++ USER INPUT +++++')
    print('host = ' + host)
    print('method = ' + method)
    print('query_type = ' + query_type)
    print('query = ' + query)

    # validate input
    validate_input(method, query_type)
```

```
   # get canonical_uri and payload
   canonical_uri, payload = get_canonical_uri_and_payload(query_type, query,
method)

   # assign payload to data or params
   data = payload if method == 'POST' else None
   params = payload if method == 'GET' else None

   # create request URL
   request_url = endpoint + canonical_uri

   # create and sign request
   creds = SimpleNamespace(
       access_key=access_key, secret_key=secret_key, token=session_token,
region=region,
   )

   request = AWSRequest(method=method, url=request_url, data=data, params=params)
   SigV4Auth(creds, service, region).add_auth(request)

   r = None

   # ************* SEND THE REQUEST *************
   if (method == 'GET'):

       print('++++ BEGIN GET REQUEST +++++')
       print('Request URL = ' + request_url)
       r = requests.get(request_url, headers=request.headers, verify=False,
params=params)

   elif (method == 'POST'):

       print('\n+++++ BEGIN POST REQUEST +++++')
       print('Request URL = ' + request_url)
       if (query_type == "loader"):
           request.headers['Content-type'] = 'application/json'
       r = requests.post(request_url, headers=request.headers, verify=False,
data=data)

   else:
       print('Request method is neither "GET" nor "POST", something is wrong
here.')

   if r is not None:
```

```
        print()
        print('+++++ RESPONSE +++++')
        print('Response code: %d\n' % r.status_code)
        response = r.text
        r.close()
        print(response)

        return response

help_msg = '''
    export AWS_ACCESS_KEY_ID=[MY_ACCESS_KEY_ID]
    export AWS_SECRET_ACCESS_KEY=[MY_SECRET_ACCESS_KEY]
    export AWS_SESSION_TOKEN=[MY_AWS_SESSION_TOKEN]
    export SERVICE_REGION=[us-east-1|us-east-2|us-west-2|eu-west-1]

    python version >=3.6 is required.

    Examples: For help
    python3 program_name.py -h

    Examples: Queries
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q status
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql/
status
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql -d
 "SELECT ?s WHERE { ?s ?p ?o }"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q sparql -d
 "SELECT ?s WHERE { ?s ?p ?o }"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q
 sparqlupdate -d "INSERT DATA { <https://s> <https://p> <https://o> }"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin/
status
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin -d
 "g.V().count()"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q gremlin -d
 "g.V().count()"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher/
status
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher
 -d "MATCH (n1) RETURN n1 LIMIT 1;"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q openCypher
 -d "MATCH (n1) RETURN n1 LIMIT 1;"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{"loadId": "68b28dcc-8e15-02b1-133d-9bd0557607e6"}'
```

```
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{}'
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q loader
 -d '{"source": "source", "format" : "csv", "failOnError": "fail_on_error",
 "iamRoleArn": "iam_role_arn", "region": "region"}'

    Environment variables must be defined as AWS_ACCESS_KEY_ID,
 AWS_SECRET_ACCESS_KEY and SERVICE_REGION.
    You should also set AWS_SESSION_TOKEN environment variable if you are using
 temporary credentials (ex. IAM Role or EC2 Instance profile).

    Current Limitations:
    - Query mode "sparqlupdate" requires POST (as per the SPARQL 1.1 protocol)
            '''

def exit_and_print_help():
    print(help_msg)
    exit()

def parse_input_and_query_neptune():


    parser = ArgumentParser(description=help_msg,
 formatter_class=RawTextHelpFormatter)
    group_host = parser.add_mutually_exclusive_group()
    group_host.add_argument("-ho", "--host", type=str)
    group_port = parser.add_mutually_exclusive_group()
    group_port.add_argument("-p", "--port", type=int, help="port ex. 8182,
 default=8182", default=8182)
    group_action = parser.add_mutually_exclusive_group()
    group_action.add_argument("-a", "--action", type=str, help="http action,
 default = GET", default="GET")
    group_endpoint = parser.add_mutually_exclusive_group()
    group_endpoint.add_argument("-q", "--query_type", type=str, help="query_type,
 default = status ", default="status")
    group_data = parser.add_mutually_exclusive_group()
    group_data.add_argument("-d", "--data", type=str, help="data required for the
 http action", default="")

    args = parser.parse_args()
    print(args)

    # Read command line parameters
    host = args.host
```

```
    port = args.port
    method = args.action
    query_type = args.query_type
    query = args.data

    if (access_key == ''):
        print('!!! ERROR: Your AWS_ACCESS_KEY_ID environment variable is
 undefined.')
        exit_and_print_help()

    if (secret_key == ''):
        print('!!! ERROR: Your AWS_SECRET_ACCESS_KEY environment variable is
 undefined.')
        exit_and_print_help()

    if (region == ''):
        print('!!! ERROR: Your SERVICE_REGION environment variable is undefined.')
        exit_and_print_help()

    if host is None:
        print('!!! ERROR: Neptune DNS is missing')
        exit_and_print_help()

    host = host + ":" + str(port)
    make_signed_request(host, method, query_type, query)


if __name__ == "__main__":
    parse_input_and_query_neptune()
```

3. Di terminal, arahkan ke lokasi file `neptunesigv4.py`.
4. Masukkan perintah berikut, ganti access key, kunci rahasia, dan Wilayah dengan nilai yang benar.

```
export AWS_ACCESS_KEY_ID=MY_ACCESS_KEY_ID
export AWS_SECRET_ACCESS_KEY=MY_SECRET_ACCESS_KEY
export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      ca-west-1 or sa-east-1 or eu-north-1 or eu-south-2 or eu-
west-1 or eu-west-2 or eu-west-3 or
                      eu-central-1 or me-south-1 or me-central-1 or il-central-1 or
 af-south-1 or ap-east-1 or
```

```
                    ap-northeast-1 or ap-northeast-2 or ap-northeast-3 or ap-
southeast-1 or ap-southeast-2 or
                    ap-southeast-3 or ap-southeast-4 or ap-southeast-5 or ap-
south-1 or ap-south-2 or
                    cn-north-1 or cn-northwest-1 or
                    us-gov-east-1 or us-gov-west-1
```

Jika Anda menggunakan kredensial sementara, Anda harus menentukan `AWS_SESSION_TOKEN` selain `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, dan `SERVICE_REGION`.

```
export AWS_SESSION_TOKEN=MY_AWS_SESSION_TOKEN
```

> **Note**
>
> Jika Anda menggunakan kredensial sementara, kredensial kedaluwarsa setelah interval tertentu, termasuk token sesinya.
> Anda harus memperbarui token sesi Anda ketika Anda meminta kredensial baru. Untuk informasi selengkapnya, lihat Menggunakan Kredensial Keamanan Sementara untuk Meminta Akses ke Sumber Daya AWS.

5. Masukkan salah satu perintah berikut untuk mengirim permintaan yang ditandatangani ke instans DB Neptune. Contoh ini menggunakan Python versi 3.6.

   Status Titik Akhir

   ```
   python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q status
   ```

   Gremlin

   ```
   python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin -d
    "g.V().count()"

   python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q gremlin -d
    "g.V().count()"
   ```

   Status Gremlin

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin/
status
```

SPARQL

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql -d
 "SELECT ?s WHERE { ?s ?p ?o }"
```

PEMBARUAN SPARQL

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q sparqlupdate
 -d "INSERT DATA { <https://s> <https://p> <https://o> }"
```

Status SPARQL

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql/status
```

OpenCypher

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher -d
 "MATCH (n1) RETURN n1 LIMIT 1;"

python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q openCypher -
d "MATCH (n1) RETURN n1 LIMIT 1;"
```

Status OpenCypher

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher/
status
```

Loader

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{"loadId": "68b28dcc-8e15-02b1-133d-9bd0557607e6"}'

python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{}'
```

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q loader
 -d '{"source": "source", "format" : "csv", "failOnError": "fail_on_error",
 "iamRoleArn": "iam_role_arn", "region": "region"}'
```

6. Sintaks untuk menjalankan skrip Python adalah sebagai berikut:

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p port -a GET|POST -q gremlin|
sparql|sparqlupdate|loader|status -d "string0data"
```

SPARQL UPDATE memerlukan POST.

Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Python Gremlin

Ikhtisar

Panduan ini menunjukkan cara menghubungkan ke database Amazon Neptunus dengan otentikasi IAM diaktifkan menggunakan driver Python Gremlin, dengan otentikasi Signature Version 4 dan SDK for Python (Boto3). AWS

Buat koneksi dasar

Gunakan contoh kode berikut sebagai panduan tentang cara membuat koneksi dasar dengan otentikasi IAM menggunakan driver Python Gremlin.

```
from boto3 import Session
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest

from gremlin_python.process.anonymous_traversal import traversal
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection

def main():
    endpoint = 'your.cluster.endpoint.neptune.amazonaws.com'
    conn_string = 'wss://' + endpoint + ':8182/gremlin'
    default_region = 'us-east-1'
    service = 'neptune-db'

    credentials = Session().get_credentials()
    if credentials is None:
```

```
        raise Exception("No AWS credentials found")
    creds = credentials.get_frozen_credentials()
    # region set inside config profile or via AWS_DEFAULT_REGION environment variable
 will be loaded
    region = Session().region_name if Session().region_name else default_region

    request = AWSRequest(method='GET', url=conn_string, data=None)
    SigV4Auth(creds, service, region).add_auth(request)

    rc = DriverRemoteConnection(conn_string, 'g', headers=request.headers.items())
    g = traversal().with_remote(rc)

    # simple query to verify connection
    count = g.V().count().next()
    print('Vertex count: ' + str(count))

    # cleanup
    rc.close()

if __name__ == "__main__":
    main()
```

Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Gremlin JavaScript

Ikhtisar

Panduan ini menunjukkan cara menyambung ke database Amazon Neptunus dengan autentikasi IAM diaktifkan menggunakan JavaScript driver Gremlin, dengan otentikasi Signature Version 4 dan SDK untuk Javascript v3. AWS

Prasyarat

- Cluster Amazon Neptunus dengan otentikasi IAM diaktifkan.
- Node 13 atau yang lebih baru (lihat versi minimal untuk Gremlin JavaScript dan AWS SDK untuk Javascript v3).
- AWS kredensyal dikonfigurasi (melalui variabel lingkungan, file kredensyal bersama, atau peran IAM).

Buat koneksi dasar

Gunakan contoh kode berikut sebagai panduan tentang cara membuat koneksi dasar dengan otentikasi IAM menggunakan driver JavaScript Gremlin.

```
const { fromNodeProviderChain } = require('@aws-sdk/credential-providers');
const { getUrlAndHeaders } = require('gremlin-aws-sigv4/lib/utils');
const { loadConfig } = require("@smithy/node-config-provider");
const { NODE_REGION_CONFIG_FILE_OPTIONS, NODE_REGION_CONFIG_OPTIONS } =
 require("@smithy/config-resolver");

const gremlin = require('gremlin');
const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;
const traversal = gremlin.process.AnonymousTraversalSource.traversal;

const DEFAULT_REGION = 'us-east-1';

const getCredentials = async () => {
    try {
        // Loads the default credential provider chain
        return await fromNodeProviderChain();
    } catch (e) {
        console.error("No credentials found", e);
        throw e;
    }
};

(main = async () => {
    console.log('Starting');

    const credentials = await getCredentials();
    try {
        // region set inside config profile or via AWS_REGION environment variable will
 be loaded
        credentials['region'] = await loadConfig(NODE_REGION_CONFIG_OPTIONS,
 NODE_REGION_CONFIG_FILE_OPTIONS)();
    } catch (e) {
        credentials['region'] = DEFAULT_REGION
    }

    const connInfo = getUrlAndHeaders(
        'you.cluster.endpoint.neptune.amazonaws.com',
        '8182',
```

```
        credentials,
        '/gremlin',
        'wss');

    const dc = new DriverRemoteConnection(connInfo['url'], {headers:
 connInfo['headers']});
    const g = traversal().withRemote(dc);

    const query = g.V().limit(5).count();
    const count = await query.next();
    console.log("Vertex count: " + count.value);

    await dc.close();
})

main();
```

Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Gremlin Go

Ikhtisar

Panduan ini menunjukkan cara menyambung ke database Amazon Neptunus dengan autentikasi IAM diaktifkan menggunakan driver Gremlin Go, dengan otentikasi Signature Version 4 dan SDK untuk GO v2. AWS

Prasyarat

- Cluster Amazon Neptunus dengan otentikasi IAM diaktifkan.
- Go 1.22 atau yang lebih baru (lihat versi minimal yang didukung untuk Gremlin Go dan SDK for AWS Go v2).
- AWS kredensil yang dikonfigurasi (melalui variabel lingkungan, file kredensial bersama, atau peran IAM)

Buat koneksi dasar

Gunakan contoh kode berikut sebagai panduan tentang cara membuat koneksi dasar dengan otentikasi IAM menggunakan driver Gremlin Go.

```
package main
```

```
import (
 "context"
 "fmt"
 "github.com/aws/aws-sdk-go-v2/config"
 "net/http"
 "strings"
 "time"

 gremlingo "github.com/apache/tinkerpop/gremlin-go/v3/driver"
 v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
)

const emptyStringSHA256 =
 `e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855`

func main() {
 neptuneEndpoint := "you.cluster.endpoint.neptune.amazonaws.com"
 connString := "wss://" + neptuneEndpoint + ":8182/gremlin"
 service := "neptune-db"
 defaultRegion := "us-east-1"

 // Create request to sign
 req, err := http.NewRequest(http.MethodGet, connString, strings.NewReader(""))
 if err != nil {
  fmt.Println(err)
  return
 }

 // Loads the default config with default credentials provider
 // See https://github.com/aws/aws-sdk-go-v2 for additional docs on API usage
 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println(err)
  return
 }
 // Retrieve loaded credentials
 cr, err := cfg.Credentials.Retrieve(context.TODO())
 if err != nil {
  fmt.Println(err)
  return
 }

 region := defaultRegion
```

```
 if cfg.Region != "" {
  // region set inside config profile, or via AWS_REGION or AWS_DEFAULT_REGION
 environment variable will be loaded
  region = cfg.Region
 }

 signer := v4.NewSigner()
 // Sign request
 err = signer.SignHTTP(context.TODO(), cr, req, emptyStringSHA256, service, "us-
east-2", time.Now())
 if err != nil {
  fmt.Println(err)
  return
 }

 // Pass the signed request header into gremlingo.HeaderAuthInfo()
 driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection(connString,
  func(settings *gremlingo.DriverRemoteConnectionSettings) {
   settings.TraversalSource = "g"
   settings.AuthInfo = gremlingo.HeaderAuthInfo(req.Header)
   // settings.TlsConfig = &tls.Config{InsecureSkipVerify: true} // Use this only if
 you're on a Mac running Go 1.18+ doing local dev. See https://github.com/golang/go/
issues/51991
  })
 if err != nil {
  fmt.Println(err)
  return
 }

 // Cleanup
 defer driverRemoteConnection.Close()

 // Creating graph traversal
 g := gremlingo.Traversal_().WithRemote(driverRemoteConnection)

 // Query execution
 count, err := g.V().Limit(5).Count().Next()
 if err != nil {
  fmt.Println(err)
  return
 }
 fmt.Println("Vertex count:", *count)
}
```

Penyegaran Kredensi Dinamis Gremlin Go

Gremlin Go memiliki DynamicAuth yang memungkinkan injeksi pointer fungsi untuk mengambil kredensil dan menghasilkan header, yang mencegah kedaluwarsa header dengan koneksi yang berjalan lama.

```
package main

import (
 "context"
 "crypto/tls"
 "fmt"
 "github.com/aws/aws-sdk-go-v2/config"
 "net/http"
 "strings"
 "time"

 gremlingo "github.com/apache/tinkerpop/gremlin-go/v3/driver"
 v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
)

const emptyStringSHA256 =
 `e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855`

func main() {
 neptuneEndpoint := "you.cluster.endpoint.neptune.amazonaws.com"
 connString := "wss://" + neptuneEndpoint + ":8182/gremlin"
 service := "neptune-db"
 defaultRegion := "us-east-1"

 //Create the request to sign
 req, err := http.NewRequest(http.MethodGet, connString, strings.NewReader(""))
 if err != nil {
  fmt.Println(err)
  return
 }

 // Loads the default config with default credentials provider
 // See https://github.com/aws/aws-sdk-go-v2 for additional docs on API usage
 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println(err)
  return
```

```
 }

 region := defaultRegion
 if cfg.Region != "" {
  // region set inside config profile, or via AWS_REGION or AWS_DEFAULT_REGION
 environment variable will be loaded
  region = cfg.Region
 }

 signer := v4.NewSigner()

 // This is the function that will be used for dynamic refreseh of credentials and
 signed headers
 gen := func() gremlingo.AuthInfoProvider {
  // Retrieve loaded credentials
  cr, err := cfg.Credentials.Retrieve(context.TODO())
  fmt.Println("AWS Credentials: ", cr)
  if err != nil {
   fmt.Println(err)
   return
  }
  // Sign request
  err = signer.SignHTTP(context.TODO(), cr, req, emptyStringSHA256, service, region,
 time.Now())
  if err != nil {
   fmt.Println(err)
   return
  }
  fmt.Println(req.Header)
  return gremlingo.HeaderAuthInfo(req.Header)
 }

 // Pass the function into gremlingo.NewDynamicAuth(), which will generate the
 AuthInfoProvider to pass into gremlingo.DriverRemoteConnectionSettings below
 auth := gremlingo.NewDynamicAuth(gen)

 driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection(connString,
  func(settings *gremlingo.DriverRemoteConnectionSettings) {
   settings.TraversalSource = "g"
   settings.AuthInfo = auth
   // settings.TlsConfig = &tls.Config{InsecureSkipVerify: true} // Use this only if
 you're on a Mac running Go 1.18+ doing local dev. See https://github.com/golang/go/
issues/51991
```

```
  })
 if err != nil {
  fmt.Println(err)
  return
 }

 // Cleanup
 defer driverRemoteConnection.Close()

 // Creating graph traversal
 g := gremlingo.Traversal_().WithRemote(driverRemoteConnection)

 // Query execution
 count, err := g.V().Limit(5).Count().Next()
 if err != nil {
  fmt.Println(err)
  return
 }
 fmt.Println("Vertex count:", *count)
}
```

Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Gremlin.NET

Ikhtisar

Panduan ini menunjukkan cara menghubungkan ke database Amazon Neptunus dengan autentikasi IAM diaktifkan menggunakan driver Gremlin.NET, dengan otentikasi Signature Version 4 dan SDK for .NET v3. AWS

Buat koneksi dasar

Untuk terhubung dengan Gremlin.NET, gunakan file `SigV4RequestSigner` sumber dari pustaka https://github.com/aws/ amazon-neptune-gremlin-dotnet kustom -sigv4. Contoh pengaturan proyek terletak di https://github.com/aws/amazon-neptune-gremlin-dotnet-sigv4/ tree/main/example, dan tercermin di bawah ini.

File Proyek:

```
<Project Sdk="Microsoft.NET.Sdk">
```

```
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp8.0</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <Compile Include="../src/SigV4RequestSigner.cs" Link="SigV4RequestSigner.cs" />
    <PackageReference Include="AWSSDK.Core" Version="3.7.402.24" />
    <PackageReference Include="gremlin.net" Version="3.7.3" />
  </ItemGroup>

</Project>
```

Contoh Program:

```
using System;
using System.Collections.Generic;
using System.Net.WebSockets;
using System.Linq;
using System.Net.Http;
using System.Security.Cryptography;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Web;
using Gremlin.Net;
using Gremlin.Net.Driver;
using Gremlin.Net.Driver.Remote;
using Gremlin.Net.Process;
using Gremlin.Net.Process.Traversal;
using Gremlin.Net.Structure;
using static Gremlin.Net.Process.Traversal.AnonymousTraversalSource;
using static Gremlin.Net.Process.Traversal.__;
using static Gremlin.Net.Process.Traversal.P;
using static Gremlin.Net.Process.Traversal.Order;
using static Gremlin.Net.Process.Traversal.Operator;
using static Gremlin.Net.Process.Traversal.Pop;
using static Gremlin.Net.Process.Traversal.Scope;
using static Gremlin.Net.Process.Traversal.TextP;
using static Gremlin.Net.Process.Traversal.Column;
using static Gremlin.Net.Process.Traversal.Direction;
using static Gremlin.Net.Process.Traversal.T;
using Amazon.Runtime.CredentialManagement;
```

```
using Amazon.Runtime;
using Amazon;
using Amazon.Util;
using Amazon.Neptune.Gremlin.Driver;

namespace NeptuneExample
{
    class Program
    {
        static void Main(string[] args)
        {
            /*
                Include your Neptune endpoint and port below.
            */
            var neptune_host = "neptune-endpoint"; // ex: mycluster.cluster.us-
east-1.neptune.amazonaws.com
            var neptune_port = 8182;

            var gremlinServer = new GremlinServer(neptune_host, neptune_port);
            var gremlinClient = new GremlinClient(gremlinServer,
 webSocketConfiguration: new SigV4RequestSigner().signRequest(neptune_host,
 neptune_port));
            var remoteConnection = new DriverRemoteConnection(gremlinClient);
            var g = Traversal().WithRemote(remoteConnection);

            /* Example code to pull the first 5 vertices in a graph. */
            Console.WriteLine("Get List of Node Labels:");
            Int32 limitValue = 5;
            var output = g.V().Limit<Vertex>(limitValue).ToList();
            foreach(var item in output) {
                Console.WriteLine(item);
            }
        }
    }
}
```

Mengelola akses ke database Amazon Neptunus menggunakan kebijakan IAM

Kebijakan IAM adalah objek JSON yang menentukan izin untuk menggunakan tindakan dan sumber daya.

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini ketika kepala sekolah membuat permintaan. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

Kebijakan Berbasis Identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada banyak identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat Pilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Menggunakan Kebijakan Kontrol Layanan (SCP) dengan organisasi AWS

Kebijakan kontrol layanan (SCPs) adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di. AWS Organizations AWS Organizations adalah layanan untuk mengelompokkan dan mengelola beberapa AWS akun secara terpusat yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam suatu organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk setiap pengguna root AWS akun. Untuk informasi selengkapnya tentang Organizations dan SCPs, lihat Cara SCPs kerja di Panduan AWS Organizations Pengguna.

Pelanggan yang menggunakan Amazon Neptunus di Akun AWS dalam Organisasi SCPs dapat memanfaatkan untuk mengontrol akun AWS mana yang dapat menggunakan Neptunus. Untuk memastikan akses ke Neptunus dalam akun anggota, pastikan untuk:

- Izinkan akses ke `rds:*` dan `neptune-db:*` untuk operasi basis data Neptunus. Lihat Mengapa izin dan sumber daya Amazon RDS diperlukan untuk menggunakan Database Neptunus? untuk detail tentang mengapa izin Amazon RDS diperlukan untuk database Neptunus.
- Izinkan akses `neptune-graph:*` untuk operasi Neptune Analytics.

Izin Diperlukan untuk Menggunakan Konsol Amazon Neptune

Agar pengguna dapat bekerja dengan konsol Amazon Neptune, pengguna tersebut harus memiliki set izin minimum. Izin ini memungkinkan pengguna untuk menjelaskan sumber daya Neptune untuk akun AWS mereka dan untuk menyediakan informasi terkait lainnya, termasuk informasi keamanan dan jaringan Amazon EC2.

Jika Anda membuat kebijakan IAM yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya bagi pengguna dengan kebijakan IAM tersebut. Untuk memastikan bahwa pengguna tersebut masih dapat menggunakan konsol Neptune, lampirkan juga kebijakan `NeptuneReadOnlyAccess` yang dikelola kepada pengguna, sebagaimana dijelaskan dalam Menggunakan kebijakan AWS terkelola untuk mengakses database Amazon Neptunus.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke API Amazon Neptunus AWS CLI atau Amazon Neptunus.

Melampirkan Kebijakan IAM ke pengguna IAM

Untuk menerapkan kebijakan terkelola atau kustom, Anda melampirkannya ke pengguna IAM. Untuk tutorial tentang topik ini, lihat Buat dan Lampirkan Kebijakan Pengelolaan Pelanggan Pertama Anda dalam Panduan Pengguna IAM.

Saat mengikuti tutorial ini, Anda dapat menggunakan salah satu contoh kebijakan yang ditunjukkan dalam bagian ini sebagai titik awal dan menyesuaikannya dengan kebutuhan Anda. Di akhir tutorial, Anda memiliki seorang pengguna IAM dengan kebijakan terlampir yang dapat menggunakan tindakan `neptune-db:*`.

> **Important**
>
> - Perubahan kebijakan IAM memakan waktu hingga 10 menit untuk diterapkan ke sumber daya Neptune yang ditentukan.

- Kebijakan IAM diterapkan ke klaster DB Neptune berlaku untuk semua instans dalam klaster itu.

Menggunakan berbagai jenis kebijakan IAM untuk mengontrol akses ke Neptunus

Untuk menyediakan akses ke tindakan administratif Neptunus atau ke data dalam klaster DB Neptunus, Anda melampirkan kebijakan ke pengguna atau peran IAM. Untuk informasi tentang cara melampirkan kebijakan IAM ke pengguna, lihatMelampirkan Kebijakan IAM ke pengguna IAM. Untuk informasi tentang melampirkan kebijakan ke peran, lihat Menambahkan dan Menghapus Kebijakan IAM dalam Panduan Pengguna IAM.

Untuk akses umum ke Neptunus, Anda dapat menggunakan salah satu kebijakan terkelola Neptunus. Untuk akses yang lebih terbatas, Anda dapat membuat kebijakan kustom Anda sendiri menggunakan tindakan administratif dan sumber daya yang didukung Neptunus..

Dalam kebijakan IAM kustom, Anda dapat menggunakan dua jenis pernyataan kebijakan berbeda yang mengontrol mode akses yang berbeda ke kluster DB Neptunus:

- Pernyataan kebijakan administratif — Pernyataan kebijakan administratif menyediakan akses ke APIs manajemen Neptunus yang Anda gunakan untuk membuat, mengonfigurasi, dan mengelola klaster DB dan instance-instance-nya.

  Karena Neptunus berbagi fungsionalitas dengan Amazon RDS, tindakan administratif, sumber daya, dan kunci kondisi dalam kebijakan Neptunus menggunakan awalan berdasarkan desain. `rds:`
- Pernyataan kebijakan akses data — Pernyataan kebijakan akses data menggunakan tindakan akses data, sumber daya, dan kunci kondisi untuk mengontrol akses data yang berisi kluster DB.

  Tindakan akses data Neptunus, sumber daya, dan kunci kondisi menggunakan awalan. `neptune-db:`

Menggunakan kunci konteks kondisi IAM di Amazon Neptunus

Anda dapat menentukan kondisi dalam pernyataan kebijakan IAM yang mengontrol akses ke Neptunus. Pernyataan kebijakan kemudian berlaku hanya jika kondisinya benar.

Misalnya, Anda mungkin ingin pernyataan kebijakan berlaku hanya setelah tanggal tertentu, atau mengizinkan akses hanya jika nilai tertentu ada dalam permintaan.

Untuk menyatakan kondisi, Anda menggunakan kunci kondisi yang telah ditentukan dalam `Condition`elemen pernyataan kebijakan, bersama dengan operator kebijakan kondisi IAM seperti sama atau kurang dari.

Jika Anda menentukan beberapa elemen `Condition` dalam sebuah pernyataan, atau beberapa kunci dalam elemen `Condition` tunggal, maka AWS akan mengevaluasinya menggunakan operasi `AND` logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan `OR` operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tanda yang sesuai dengan nama pengguna IAM mereka. Untuk informasi lebih lanjut, lihat Elemen Kebijakan IAM: Variabel dan Tanda dalam Panduan Pengguna IAM.

Tipe data dari kunci kondisi menentukan operator kondisi yang dapat Anda gunakan untuk membandingkan nilai dalam permintaan dengan nilai dalam pernyataan kebijakan. Jika Anda menggunakan operator kondisi yang tidak kompatibel dengan tipe data tersebut, kecocokan selalu gagal dan pernyataan kebijakan tidak pernah berlaku.

Neptunus mendukung kumpulan kunci kondisi yang berbeda untuk pernyataan kebijakan administratif daripada untuk pernyataan kebijakan akses data:

- Kunci kondisi untuk pernyataan kebijakan administratif
- Kunci kondisi untuk pernyataan kebijakan akses data

Dukungan untuk kebijakan IAM dan fitur kontrol akses di Amazon Neptunus

Tabel berikut menunjukkan fitur IAM yang didukung Neptunus untuk pernyataan kebijakan administratif dan pernyataan kebijakan akses data:

Fitur IAM yang dapat Anda gunakan dengan Neptunus

Fitur IAM	Administratif	Akses data
Kebijakan berbasis identitas	Ya	Ya

Fitur IAM	Administratif	Akses data
Kebijakan berbasis sumber daya	Tidak	Tidak
Tindakan kebijakan	Ya	Ya
Sumber daya kebijakan	Ya	Ya
Kunci kondisi global	Ya	(subset)
Kunci kondisi berbasis tag	Ya	Tidak
Daftar Kontrol Akses (ACLs)	Tidak	Tidak
Kebijakan kontrol layanan (SCPs)	Ya	Ya
Peran terkait layanan	Ya	Tidak

Keterbatasan Kebijakan IAM

Perubahan kebijakan IAM memakan waktu hingga 10 menit untuk diterapkan ke sumber daya Neptune yang ditentukan.

Kebijakan IAM diterapkan ke klaster DB Neptune berlaku untuk semua instans dalam klaster itu.

Neptunus saat ini tidak mendukung kontrol akses lintas akun di tingkat bidang data. Kontrol akses lintas akun hanya didukung saat pemuatan massal dan dengan menggunakan rantai peran. Untuk informasi lebih lanjut, lihat Tutorial pemuatan massal.

Menggunakan kebijakan AWS terkelola untuk mengakses database Amazon Neptunus

AWS mengatasi banyak kasus penggunaan umum dengan menyediakan kebijakan IAM mandiri yang dibuat dan dikelola oleh. AWS Kebijakan terkelola memberikan izin yang diperlukan untuk kasus penggunaan umum sehingga Anda tidak perlu menyelidiki izin apa yang diperlukan. Untuk informasi selengkapnya, lihat Kebijakan Terkelola AWS dalam Panduan Pengguna IAM.

Kebijakan AWS terkelola berikut, yang dapat Anda lampirkan ke pengguna di akun Anda, adalah untuk menggunakan manajemen Amazon Neptunus APIs:

- NeptuneReadOnlyAccess— Memberikan akses hanya-baca ke semua sumber daya Neptunus untuk tujuan administratif dan akses data di akun root. AWS
- NeptuneFullAccess— Memberikan akses penuh ke semua sumber daya Neptunus untuk tujuan administratif dan akses data di akun root. AWS Ini disarankan jika Anda memerlukan akses Neptunus penuh dari atau SDK, AWS CLI tetapi tidak untuk akses. Konsol Manajemen AWS
- NeptuneConsoleFullAccess— Memberikan akses penuh di AWS akun root ke semua tindakan dan sumber daya administratif Neptunus, tetapi tidak untuk tindakan atau sumber daya akses data apa pun. Ini juga mencakup izin tambahan untuk menyederhanakan akses Neptunus dari konsol, termasuk izin IAM dan Amazon EC2 (VPC) terbatas.
- NeptuneGraphReadOnlyAccess — Menyediakan akses hanya-baca ke semua sumber daya Amazon Neptunus Analytics bersama dengan izin hanya-baca untuk layanan dependen
- AWSServiceRoleForNeptuneGraphPolicy— Memungkinkan grafik Neptunus Analytics untuk CloudWatch mempublikasikan metrik dan log operasional dan penggunaan.

Peran dan kebijakan IAM Neptune memberikan beberapa akses ke sumber daya Amazon RDS, karena Neptune berbagi teknologi operasional dengan Amazon RDS untuk fitur manajemen tertentu. Ini termasuk izin API administratif, itulah sebabnya tindakan administratif Neptunus memiliki awalan. `rds:`

Pembaruan kebijakan terkelola AWS Neptunus

Tabel berikut melacak pembaruan kebijakan terkelola Neptunus mulai dari saat Neptunus mulai melacak perubahan ini:

Kebijakan	Deskripsi	Date
AWS kebijakan terkelola untuk Amazon Neptunus - perbarui ke kebijakan yang ada	Kebijakan `NeptuneReadOnlyAcess` dan `NeptuneFullAccess` terkelola sekarang menyertakan `Sid` (ID pernyataan) sebagai pengenal dalam pernyataan kebijakan.	2024-01-22

Kebijakan	Deskripsi	Date
NeptuneGraphReadOnlyAccess(dirilis)	Dirilis untuk menyediakan akses hanya-baca ke grafik dan sumber daya Neptunus Analytics.	2023-11-29
AWSServiceRoleForNeptuneGraphPolicy(dirilis)	Dirilis untuk memungkinkan akses grafik Neptunus Analytics CloudWatch untuk mempublikasikan metrik dan log operasional dan penggunaan. Lihat Menggunakan peran terkait layanan (SLRs) di Neptunus Analytics.	2023-11-29
NeptuneConsoleFullAccess(izin tambahan)	Izin tambahan menyediakan semua akses yang diperlukan untuk berinteraksi dengan grafik Neptunus Analytics.	2023-11/29
NeptuneFullAccess(izin tambahan)	Menambahkan izin akses data, dan izin untuk database global baru. APIs	2022-07-28
NeptuneConsoleFullAccess(izin tambahan)	Menambahkan izin untuk database APIs global baru.	2022-07-21
Neptunus mulai melacak perubahan	Neptunus mulai melacak perubahan kebijakan yang dikelola. AWS	2022-07-21

Pemberian database **NeptuneReadOnlyAccess** Amazon Neptunus menggunakan kebijakan terkelola AWS

Kebijakan NeptuneReadOnlyAccessterkelola di bawah ini memberikan akses hanya-baca ke semua tindakan dan sumber daya Neptunus untuk tujuan administratif dan akses data.

> **Note**
>
> Kebijakan ini diperbarui pada 2022-07-21 untuk menyertakan izin akses data hanya-baca serta izin administratif hanya-baca dan menyertakan izin untuk tindakan database global.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowReadOnlyPermissionsForRDS",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DownloadDBLogFilePortion",
```

```
                "rds:ListTagsForResource"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForCloudwatch",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForEC2",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForKMS",
            "Effect": "Allow",
            "Action": [
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "kms:ListAliases",
                "kms:ListKeyPolicies"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForLogs",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams",
                "logs:GetLogEvents"
```

```
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*",
                "arn:aws:logs:*:*:log-group:/aws/neptune/*:log-stream:*"
            ]
        },
        {
            "Sid": "AllowReadOnlyPermissionsForNeptuneDB",
            "Effect": "Allow",
            "Action": [
                "neptune-db:Read*",
                "neptune-db:Get*",
                "neptune-db:List*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Pemberian database **NeptuneFullAccess** Amazon Neptunus menggunakan kebijakan terkelola AWS

Kebijakan NeptuneFullAccessterkelola di bawah ini memberikan akses penuh ke semua tindakan dan sumber daya Neptunus untuk tujuan administratif dan akses data. Disarankan jika Anda memerlukan akses penuh dari AWS CLI atau dari SDK, tetapi tidak dari Konsol Manajemen AWS.

> **Note**
>
> Kebijakan ini diperbarui pada 2022-07-21 untuk menyertakan izin akses data lengkap serta izin administratif penuh dan untuk menyertakan izin untuk tindakan database global.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AllowNeptuneCreate",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBCluster",
                "rds:CreateDBInstance"
            ],
            "Resource": [
                "arn:aws:rds:*:*:*"
            ],
            "Condition": {
                "StringEquals": {
                    "rds:DatabaseEngine": [
                        "graphdb",
                        "neptune"
                    ]
                }
            }
        },
        {
            "Sid": "AllowManagementPermissionsForRDS",
            "Effect": "Allow",
            "Action": [
                "rds:AddRoleToDBCluster",
                "rds:AddSourceIdentifierToSubscription",
                "rds:AddTagsToResource",
                "rds:ApplyPendingMaintenanceAction",
                "rds:CopyDBClusterParameterGroup",
                "rds:CopyDBClusterSnapshot",
                "rds:CopyDBParameterGroup",
                "rds:CreateDBClusterEndpoint",
                "rds:CreateDBClusterParameterGroup",
                "rds:CreateDBClusterSnapshot",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSubnetGroup",
                "rds:CreateEventSubscription",
                "rds:CreateGlobalCluster",
                "rds:DeleteDBCluster",
                "rds:DeleteDBClusterEndpoint",
                "rds:DeleteDBClusterParameterGroup",
                "rds:DeleteDBClusterSnapshot",
                "rds:DeleteDBInstance",
                "rds:DeleteDBParameterGroup",
                "rds:DeleteDBSubnetGroup",
                "rds:DeleteEventSubscription",
```

```
                "rds:DeleteGlobalCluster",
                "rds:DescribeDBClusterEndpoints",
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEngineDefaultClusterParameters",
                "rds:DescribeEngineDefaultParameters",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOptionGroups",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DescribeValidDBInstanceModifications",
                "rds:DownloadDBLogFilePortion",
                "rds:FailoverDBCluster",
                "rds:FailoverGlobalCluster",
                "rds:ListTagsForResource",
                "rds:ModifyDBCluster",
                "rds:ModifyDBClusterEndpoint",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyDBSubnetGroup",
                "rds:ModifyEventSubscription",
                "rds:ModifyGlobalCluster",
                "rds:PromoteReadReplicaDBCluster",
                "rds:RebootDBInstance",
                "rds:RemoveFromGlobalCluster",
                "rds:RemoveRoleFromDBCluster",
                "rds:RemoveSourceIdentifierFromSubscription",
```

```
                "rds:RemoveTagsFromResource",
                "rds:ResetDBClusterParameterGroup",
                "rds:ResetDBParameterGroup",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:RestoreDBClusterToPointInTime",
                "rds:StartDBCluster",
                "rds:StopDBCluster"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "AllowOtherDepedentPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs",
                "kms:ListAliases",
                "kms:ListKeyPolicies",
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "sns:ListSubscriptions",
                "sns:ListTopics",
                "sns:Publish"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "AllowPassRoleForNeptune",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {
                    "iam:passedToService": "rds.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowCreateSLRForNeptune",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "rds.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowDataAccessForNeptune",
            "Effect": "Allow",
            "Action": [
                "neptune-db:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Pemberian **`NeptuneConsoleFullAccess`** menggunakan kebijakan AWS terkelola

Kebijakan NeptuneConsoleFullAccessterkelola di bawah ini memberikan akses penuh ke semua tindakan dan sumber daya Neptunus untuk tujuan administratif, tetapi tidak untuk tujuan akses data. Ini juga mencakup izin tambahan untuk menyederhanakan akses Neptunus dari konsol, termasuk izin IAM dan Amazon EC2 (VPC) terbatas.

> **Note**
>
> Kebijakan ini diperbarui pada 2023-11-29 untuk menyertakan izin yang diperlukan untuk berinteraksi dengan grafik Neptunus Analytics.

Itu diperbarui pada 2022-07-21 untuk menyertakan izin untuk tindakan database global.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowNeptuneCreate",
      "Effect": "Allow",
      "Action": [
        "rds:CreateDBCluster",
        "rds:CreateDBInstance"
      ],
      "Resource": [
        "arn:aws:rds:*:*:*"
      ],
      "Condition": {
        "StringEquals": {
          "rds:DatabaseEngine": [
            "graphdb",
            "neptune"
          ]
        }
      }
    },
    {
      "Sid": "AllowManagementPermissionsForRDS",
      "Action": [
        "rds:AddRoleToDBCluster",
        "rds:AddSourceIdentifierToSubscription",
        "rds:AddTagsToResource",
        "rds:ApplyPendingMaintenanceAction",
        "rds:CopyDBClusterParameterGroup",
        "rds:CopyDBClusterSnapshot",
        "rds:CopyDBParameterGroup",
        "rds:CreateDBClusterParameterGroup",
        "rds:CreateDBClusterSnapshot",
        "rds:CreateDBParameterGroup",
        "rds:CreateDBSubnetGroup",
        "rds:CreateEventSubscription",
```

```
        "rds:DeleteDBCluster",
        "rds:DeleteDBClusterParameterGroup",
        "rds:DeleteDBClusterSnapshot",
        "rds:DeleteDBInstance",
        "rds:DeleteDBParameterGroup",
        "rds:DeleteDBSubnetGroup",
        "rds:DeleteEventSubscription",
        "rds:DescribeAccountAttributes",
        "rds:DescribeCertificates",
        "rds:DescribeDBClusterParameterGroups",
        "rds:DescribeDBClusterParameters",
        "rds:DescribeDBClusterSnapshotAttributes",
        "rds:DescribeDBClusterSnapshots",
        "rds:DescribeDBClusters",
        "rds:DescribeDBEngineVersions",
        "rds:DescribeDBInstances",
        "rds:DescribeDBLogFiles",
        "rds:DescribeDBParameterGroups",
        "rds:DescribeDBParameters",
        "rds:DescribeDBSecurityGroups",
        "rds:DescribeDBSubnetGroups",
        "rds:DescribeEngineDefaultClusterParameters",
        "rds:DescribeEngineDefaultParameters",
        "rds:DescribeEventCategories",
        "rds:DescribeEventSubscriptions",
        "rds:DescribeEvents",
        "rds:DescribeOptionGroups",
        "rds:DescribeOrderableDBInstanceOptions",
        "rds:DescribePendingMaintenanceActions",
        "rds:DescribeValidDBInstanceModifications",
        "rds:DownloadDBLogFilePortion",
        "rds:FailoverDBCluster",
        "rds:ListTagsForResource",
        "rds:ModifyDBCluster",
        "rds:ModifyDBClusterParameterGroup",
        "rds:ModifyDBClusterSnapshotAttribute",
        "rds:ModifyDBInstance",
        "rds:ModifyDBParameterGroup",
        "rds:ModifyDBSubnetGroup",
        "rds:ModifyEventSubscription",
        "rds:PromoteReadReplicaDBCluster",
        "rds:RebootDBInstance",
        "rds:RemoveRoleFromDBCluster",
        "rds:RemoveSourceIdentifierFromSubscription",
```

```
        "rds:RemoveTagsFromResource",
        "rds:ResetDBClusterParameterGroup",
        "rds:ResetDBParameterGroup",
        "rds:RestoreDBClusterFromSnapshot",
        "rds:RestoreDBClusterToPointInTime"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowOtherDepedentPermissions",
      "Action": [
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:ListMetrics",
        "ec2:AllocateAddress",
        "ec2:AssignIpv6Addresses",
        "ec2:AssignPrivateIpAddresses",
        "ec2:AssociateAddress",
        "ec2:AssociateRouteTable",
        "ec2:AssociateSubnetCidrBlock",
        "ec2:AssociateVpcCidrBlock",
        "ec2:AttachInternetGateway",
        "ec2:AttachNetworkInterface",
        "ec2:CreateCustomerGateway",
        "ec2:CreateDefaultSubnet",
        "ec2:CreateDefaultVpc",
        "ec2:CreateInternetGateway",
        "ec2:CreateNatGateway",
        "ec2:CreateNetworkInterface",
        "ec2:CreateRoute",
        "ec2:CreateRouteTable",
        "ec2:CreateSecurityGroup",
        "ec2:CreateSubnet",
        "ec2:CreateVpc",
        "ec2:CreateVpcEndpoint",
        "ec2:CreateVpcEndpoint",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAddresses",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeCustomerGateways",
```

```
        "ec2:DescribeInstances",
        "ec2:DescribeNatGateways",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribePrefixLists",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroupReferences",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs",
        "ec2:DescribeVpcs",
        "ec2:ModifyNetworkInterfaceAttribute",
        "ec2:ModifySubnetAttribute",
        "ec2:ModifyVpcAttribute",
        "ec2:ModifyVpcEndpoint",
        "iam:ListRoles",
        "kms:ListAliases",
        "kms:ListKeyPolicies",
        "kms:ListKeys",
        "kms:ListRetirableGrants",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents",
        "sns:ListSubscriptions",
        "sns:ListTopics",
        "sns:Publish"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowPassRoleForNeptune",
      "Action": "iam:PassRole",
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:passedToService": "rds.amazonaws.com"
        }
```

```
      }
    },
    {
      "Sid": "AllowCreateSLRForNeptune",
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "rds.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowManagementPermissionsForNeptuneAnalytics",
      "Effect": "Allow",
      "Action": [
        "neptune-graph:CreateGraph",
        "neptune-graph:DeleteGraph",
        "neptune-graph:GetGraph",
        "neptune-graph:ListGraphs",
        "neptune-graph:UpdateGraph",
        "neptune-graph:ResetGraph",
        "neptune-graph:CreateGraphSnapshot",
        "neptune-graph:DeleteGraphSnapshot",
        "neptune-graph:GetGraphSnapshot",
        "neptune-graph:ListGraphSnapshots",
        "neptune-graph:RestoreGraphFromSnapshot",
        "neptune-graph:CreatePrivateGraphEndpoint",
        "neptune-graph:GetPrivateGraphEndpoint",
        "neptune-graph:ListPrivateGraphEndpoints",
        "neptune-graph:DeletePrivateGraphEndpoint",
        "neptune-graph:CreateGraphUsingImportTask",
        "neptune-graph:GetImportTask",
        "neptune-graph:ListImportTasks",
        "neptune-graph:CancelImportTask"
      ],
      "Resource": [
        "arn:aws:neptune-graph:*:*:*"
      ]
    },
    {
      "Sid": "AllowPassRoleForNeptuneAnalytics",
```

```
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:passedToService": "neptune-graph.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowCreateSLRForNeptuneAnalytics",
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::*:role/aws-service-role/neptune-
graph.amazonaws.com/AWSServiceRoleForNeptuneGraph",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "neptune-graph.amazonaws.com"
        }
      }
    }
  ]
}
```

Pemberian **`NeptuneGraphReadOnlyAccess`** menggunakan kebijakan AWS terkelola

Kebijakan NeptuneGraphReadOnlyAccessterkelola di bawah ini menyediakan akses hanya baca ke semua sumber daya Amazon Neptunus Analytics bersama dengan izin baca saja untuk layanan dependen.

Kebijakan ini mencakup izin untuk melakukan hal berikut:

- Untuk Amazon EC2 — Ambil informasi tentang VPCs, subnet, grup keamanan, dan zona ketersediaan.
- Untuk AWS KMS - Ambil informasi tentang kunci dan alias KMS.
- Untuk CloudWatch — Ambil informasi tentang CloudWatch metrik.
- Untuk CloudWatch Log - Ambil informasi tentang aliran CloudWatch log dan peristiwa.

> **Note**
>
> Kebijakan ini dirilis pada 2023-11-29.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowReadOnlyPermissionsForNeptuneGraph",
      "Effect": "Allow",
      "Action": [
        "neptune-graph:Get*",
        "neptune-graph:List*",
        "neptune-graph:Read*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForEC2",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:DescribeAvailabilityZones"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForKMS",
      "Effect": "Allow",
      "Action": [
        "kms:ListKeys",
        "kms:ListAliases"
      ],
      "Resource": "*"
    },
```

```
    {
      "Sid": "AllowReadOnlyPermissionsForCloudwatch",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:GetMetricData",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForLogs",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams",
        "logs:GetLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/neptune/*:log-stream:*"
      ]
    }
  ]
}
```

Memberikan akses ke grafik Neptunus menggunakan **`AWSServiceRoleForNeptuneGraphPolicy`**

Kebijakan AWSServiceRoleForNeptuneGraphPolicyterkelola di bawah ini memberikan akses grafik CloudWatch untuk mempublikasikan metrik dan log operasional dan penggunaan. Lihat nan-service-linked-roles.

> **Note**
>
> Kebijakan ini dirilis pada 2023-11-29.

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "GraphMetrics",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": [
            "AWS/Neptune",
            "AWS/Usage"
          ]
        }
      }
    },
    {
      "Sid": "GraphLogGroup",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/neptune/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
      }
    },
    {
      "Sid": "GraphLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/neptune/*:log-stream:*"
      ],
      "Condition": {
```

```
        "StringEquals": {
          "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
      }
    }
  ]
}
```

Menyesuaikan akses ke sumber daya Amazon Neptunus menggunakan kunci konteks kondisi IAM

Anda dapat menentukan kondisi dalam kebijakan IAM yang mengontrol akses ke tindakan dan sumber daya manajemen Neptunus. Pernyataan kebijakan kemudian berlaku hanya jika kondisinya benar.

Misalnya, Anda mungkin ingin pernyataan kebijakan berlaku hanya setelah tanggal tertentu, atau mengizinkan akses hanya jika nilai tertentu ada dalam permintaan API.

Untuk menyatakan kondisi, Anda menggunakan kunci kondisi yang telah ditentukan dalam `Condition`elemen pernyataan kebijakan, bersama dengan operator kebijakan kondisi IAM seperti sama atau kurang dari.

Jika Anda menentukan beberapa elemen `Condition` dalam sebuah pernyataan, atau beberapa kunci dalam elemen `Condition` tunggal, maka AWS akan mengevaluasinya menggunakan operasi `AND` logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan `OR` operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tanda yang sesuai dengan nama pengguna IAM mereka. Untuk informasi lebih lanjut, lihat Elemen Kebijakan IAM: Variabel dan Tanda dalam Panduan Pengguna IAM.

Tipe data dari kunci kondisi menentukan operator kondisi yang dapat Anda gunakan untuk membandingkan nilai dalam permintaan dengan nilai dalam pernyataan kebijakan. Jika Anda menggunakan operator kondisi yang tidak kompatibel dengan tipe data tersebut, kecocokan selalu gagal dan pernyataan kebijakan tidak pernah berlaku.

Kunci kondisi IAM untuk pernyataan kebijakan administratif Neptunus

- Kunci kondisi global - Anda dapat menggunakan sebagian besar kunci kondisi AWS global dalam pernyataan kebijakan administratif Neptunus.
- Kunci kondisi khusus layanan — Ini adalah kunci yang ditentukan untuk layanan tertentu AWS . Yang didukung Neptunus untuk pernyataan kebijakan administratif tercantum dalam. Kunci kondisi IAM untuk mengelola Amazon Neptunus

Kunci kondisi IAM untuk pernyataan kebijakan akses data Neptunus

- Kunci kondisi global - Subset dari kunci ini yang didukung Neptunus dalam pernyataan kebijakan akses data tercantum dalam. AWS kunci konteks kondisi global yang didukung oleh Neptunus dalam pernyataan kebijakan akses data
- Kunci kondisi khusus layanan yang didefinisikan Neptunus untuk pernyataan kebijakan akses data dicantumkan di dalamnya. Kunci kondisi

Membuat pernyataan kebijakan IAM khusus untuk mengelola Amazon Neptunus

Pernyataan kebijakan administratif memungkinkan Anda mengontrol apa yang dapat dilakukan pengguna IAM untuk mengelola database Neptunus.

Pernyataan kebijakan administratif Neptunus memberikan akses ke satu atau lebih tindakan administratif dan sumber daya administratif yang didukung Neptunus. Anda juga dapat menggunakan Kunci kondisi untuk membuat izin administratif lebih spesifik.

> **Note**
>
> Karena Neptunus berbagi fungsionalitas dengan Amazon RDS, tindakan administratif, sumber daya, dan kunci kondisi khusus layanan dalam pernyataan kebijakan administratif menggunakan awalan berdasarkan desain. `rds:`

Topik

- Tindakan IAM untuk mengelola Amazon Neptunus
- Jenis sumber daya IAM untuk mengelola Amazon Neptunus

- Kunci kondisi IAM untuk mengelola Amazon Neptunus
- Membuat pernyataan kebijakan administratif IAM untuk Amazon Neptunus

Tindakan IAM untuk mengelola Amazon Neptunus

Anda dapat menggunakan tindakan administratif yang tercantum di bawah ini dalam `Action` elemen pernyataan kebijakan IAM untuk mengontrol akses ke manajemen Neptunus. APIs Saat Anda menggunakan sebuah tindakan dalam sebuah kebijakan, Anda biasanya mengizinkan atau menolak akses ke operasi API atau perintah CLI dengan nama yang sama. Namun, dalam beberapa kasus, satu tindakan tunggal mengontrol akses ke lebih dari satu operasi. Atau, beberapa operasi memerlukan beberapa tindakan yang berbeda.

`Resource type`Bidang dalam daftar di bawah ini menunjukkan apakah setiap tindakan mendukung izin tingkat sumber daya. Jika tidak ada nilai di bidang ini, Anda harus menentukan semua sumber daya ("*") dalam `Resource` elemen pernyataan kebijakan Anda. Jika kolom menyertakan jenis sumber daya, maka Anda dapat menentukan ARN sumber daya dari jenis itu dalam pernyataan dengan tindakan itu. Jenis sumber daya administratif Neptunus tercantum di halaman ini.

Sumber daya yang diperlukan ditunjukkan dalam daftar di bawah ini dengan tanda bintang (*). Jika Anda menentukan ARN izin tingkat sumber daya dalam pernyataan dengan menggunakan tindakan ini, maka izinnya harus jenis ini. Beberapa tindakan mendukung berbagai jenis sumber daya. Jika jenis sumber daya adalah opsional (dengan kata lain, tidak ditandai dengan tanda bintang), maka Anda tidak perlu memasukkannya.

Untuk informasi selengkapnya tentang bidang yang tercantum di sini, lihat tabel tindakan di Panduan Pengguna IAM.

rds: AddRoleTo DBCluster

`AddRoleToDBCluster` mengaitkan IAM role dengan klaster DB Neptune.

Tingkat akses: `Write`.

Tindakan bergantung: `iam:PassRole`.

Jenis sumber daya: klaster (wajib).

rds: AddSourceIdentifierToSubscription

`AddSourceIdentifierToSubscription` menambahkan pengidentifikasi sumber ke langganan notifikasi Neptune yang ada.

Tingkat akses: `Write`.

Jenis sumber daya: es (wajib).

rds: AddTagsToResource

`AddTagsToResource` mengaitkan IAM role dengan klaster DB Neptune.

Tingkat akses: `Write`.

Jenis sumber daya:

- db
- es
- pg
- cuplikan cluster
- subgrp

Kunci Kondisi:

- aws:RequestTag/*tag-key*
- aws: TagKeys

rds: ApplyPendingMaintenanceAction

`ApplyPendingMaintenanceAction` menerapkan tindakan pemeliharaan tertunda ke sumber daya.

Tingkat akses: `Write`.

Jenis sumber daya: db (wajib).

RDS: Salin DBCluster ParameterGroup

`Salin DBCluster ParameterGroup` menyalin grup parameter klaster DB yang ditentukan.

Tingkat akses: `Write`.

Jenis sumber daya: cluster-pg (wajib).

RDSDBCluster: Salin Snapshot

`Salin DBCluster Snapshot` menyalin snapshot dari klaster DB.

Tingkat akses: `Write`.

Jenis sumber daya: cluster-snapshot (wajib).

RDSDBParameter: Grup Salin

`Salin DBParameter Grup` menyalin grup parameter DB yang ditentukan.

Tingkat akses: `Write`.

Jenis sumber daya: pg (wajib).

RDS: Buat DBCluster

`Buat DBCluster` menciptakan sebuah klaster DB Neptune baru.

Tingkat akses: `Tagging`.

Tindakan bergantung: `iam:PassRole`.

Jenis sumber daya:

- cluster (wajib).
- cluster-pg (wajib).
- subgrp (wajib).

Kunci Kondisi:

- aws:RequestTag/*tag-key*
- aws: TagKeys
- neptunus-rds_ DatabaseEngine

RDS: Buat DBCluster ParameterGroup

`Buat DBCluster ParameterGroup` membuat grup parameter klaster DB baru.

Tingkat akses: `Tagging`.

Jenis sumber daya: cluster-pg (wajib).

Kunci Kondisi:

- aws:RequestTag/*tag-key*
- aws: TagKeys

RDSDBCluster: Buat Snapshot

`Buat DBCluster Snapshot` membuat snapshot klaster DB.

Tingkat akses: `Tagging`.

Jenis sumber daya:

- cluster (wajib).
- cluster-snapshot (wajib).

Kunci Kondisi:

- aws:RequestTag/*tag-key*
- aws: TagKeys

RDS: Buat DBInstance

`Buat DBInstance` menciptakan instans DB baru.

Tingkat akses: `Tagging`.

Tindakan bergantung: `iam:PassRole`.

Jenis sumber daya:

- db (wajib).
- pg (wajib).
- subgrp (wajib).

Kunci Kondisi:

- aws:RequestTag/*tag-key*
- aws: TagKeys

RDSDBParameter: Buat Grup

`Buat DBParameter Grup` membuat grup parameter DB baru.

Tingkat akses: `Tagging`.

Jenis sumber daya: pg (wajib).

Kunci Kondisi:

- aws:RequestTag/*tag-key*
- aws: TagKeys

RDSDBSubnet: Buat Grup

`Buat DBSubnet Grup` membuat grup subnet DB baru.

Tingkat akses: `Tagging`.

Jenis sumber daya: subgrp (wajib).

Kunci Kondisi:

- aws:RequestTag/*tag-key*
- aws: TagKeys

rds: CreateEventSubscription

`CreateEventSubscription` menciptakan langganan notifikasi peristiwa Neptune.

Tingkat akses: `Tagging`.

Jenis sumber daya: es (wajib).

Kunci Kondisi:

- aws:RequestTag/*tag-key*
- aws: TagKeys

RDS: Hapus DBCluster

`Hapus DBCluster` menghapus klaster DB Neptune yang ada.

Tingkat akses: `Write`.

Jenis sumber daya:

- cluster (wajib).
- cluster-snapshot (wajib).

RDS: Hapus DBCluster ParameterGroup

`Hapus DBCluster ParameterGroup` menghapus grup parameter klaster DB yang ditentukan.

Tingkat akses: `Write`.

Jenis sumber daya: cluster-pg (wajib).

RDSDBCluster: Hapus Snapshot

`Hapus DBCluster Snapshot` menghapus snapshot klaster DB.

Tingkat akses: `Write`.

Jenis sumber daya: cluster-snapshot (wajib).

RDS: Hapus DBInstance

`Hapus DBInstance` menghapus instans DB yang ditentukan.

Tingkat akses: `Write`.

Jenis sumber daya: db (wajib).

RDSDBParameter: Hapus Grup

`Hapus DBParameter Grup`menghapus DBParameter Grup tertentu.

Tingkat akses: `Write`.

Jenis sumber daya: pg (wajib).

RDSDBSubnet: Hapus Grup

`Hapus DBSubnet Grup` menghapus grup subnet DB.

Tingkat akses: `Write`.

Jenis sumber daya: subgrp (wajib).

rds: DeleteEventSubscription

`DeleteEventSubscription` menghapus langganan pemberitahuan kejadian.

Tingkat akses: `Write`.

Jenis sumber daya: es (wajib).

RDS: jelaskan DBCluster ParameterGroups

`Jelaskan DBCluster ParameterGroups`mengembalikan daftar DBCluster ParameterGroup deskripsi.

Tingkat akses: `List`.

Jenis sumber daya: cluster-pg (wajib).

RDSDBCluster: Jelaskan Parameter

`Jelaskan DBCluster Parameter` mengembalikan daftar parameter detail untuk grup parameter kluster DB tertentu.

Tingkat akses: `List`.

Jenis sumber daya: cluster-pg (wajib).

RDS: jelaskan DBCluster SnapshotAttributes

`Jelaskan DBCluster SnapshotAttributes` mengembalikan daftar nama atribut dan nilao snapshot klaster DB untuk snapshot klaster DB manual.

Tingkat akses: `List`.

Jenis sumber daya: cluster-snapshot (wajib).

RDSDBCluster: Jelaskan Snapshots

`Jelaskan DBCluster Snapshots` mengembalikan informasi tentang snapshot klaster DB.

Tingkat akses: `Read`.

RDS: jelaskan DBClusters

`Jelaskan DBClusters` mengembalikan informasi tentang klaster DB Neptune yang disediakan.

Tingkat akses: `List`.

Jenis sumber daya: klaster (wajib).

RDSDBEngine: Jelaskan Versi

`Jelaskan DBEngine Versi` mengembalikan daftar mesin DB yang tersedia.

Tingkat akses: `List`.

Jenis sumber daya: pg (wajib).

RDS: jelaskan DBInstances

`Jelaskan DBInstances` mengembalikan informasi tentang instans DB.

Tingkat akses: `List`.

Jenis sumber daya: es (wajib).

RDSDBParameter: Jelaskan Grup

`Jelaskan DBParameter Grup`mengembalikan daftar deskripsi DBParameter Grup.

Tingkat akses: `List`.

Jenis sumber daya: pg (wajib).

RDS: jelaskan DBParameters

`Jelaskan DBParameters` mengembalikan daftar parameter detail untuk grup parameter DB tertentu.

Tingkat akses: `List`.

Jenis sumber daya: pg (wajib).

RDSDBSubnet: Jelaskan Grup

`Jelaskan DBSubnet Grup`mengembalikan daftar deskripsi DBSubnet Grup.

Tingkat akses: `List`.

Jenis sumber daya: subgrp (wajib).

rds: DescribeEventCategories

`DescribeEventCategories` mengembalikan daftar kategori untuk semua jenis sumber peristiwa, atau, jika ditentukan, untuk jenis sumber yang tertentu.

Tingkat akses: `List`.

rds: DescribeEventSubscriptions

`DescribeEventSubscriptions` mencantumkan semua deskripsi langganan untuk akun pelanggan.

Tingkat akses: `List`.

Jenis sumber daya: es (wajib).

rds: DescribeEvents

`DescribeEvents` mengembalikan peristiwa yang terkait dengan instans DB, grup keamanan DB, dan grup parameter DB selama 14 hari terakhir.

Tingkat akses: `List`.

Jenis sumber daya: es (wajib).

rds: Pilihan DescribeOrderable DBInstance

`DescribeOrderableDBInstancePilihan` mengembalikan daftar opsi instans DB yang dapat diurutkan untuk mesin yang ditentukan.

Tingkat akses: `List`.

rds: DescribePendingMaintenanceActions

`DescribePendingMaintenanceActions` mengembalikan daftar sumber daya (misalnya, instans DB) yang memiliki setidaknya satu tindakan pemeliharaan tertunda.

Tingkat akses: `List`.

Jenis sumber daya: db (wajib).

rds: DescribeValid DBInstance Modifikasi

`DescribeValidDBInstanceModifikasi` mencantumkan modifikasi yang tersedia yang dapat Anda buat pada instans DB Anda.

Tingkat akses: `List`.

Jenis sumber daya: db (wajib).

RDS: Failover DBCluster

`Kegagalan DBCluster` memaksakan failover untuk klaster DB.

Tingkat akses: `Write`.

Jenis sumber daya: klaster (wajib).

rds: ListTagsForResource

`ListTagsForResource` mencantumkan semua tanda pada sumber daya Neptune.

Tingkat akses: `Read`.

Jenis sumber daya:

- cuplikan cluster
- db
- es
- pg
- subgrp

RDS: memodifikasi DBCluster

`Memodifikasi DBCluster`

Memodifikasi pengaturan untuk klaster DB Neptune.

Tingkat akses: `Write`.

Tindakan bergantung: `iam:PassRole`.

Jenis sumber daya:

- cluster (wajib).
- cluster-pg (wajib).

RDS: memodifikasi DBCluster ParameterGroup

`Memodifikasi DBCluster ParameterGroup` mengubah parameter grup parameter klaster DB.

Tingkat akses: `Write`.

Jenis sumber daya: cluster-pg (wajib).

RDS: memodifikasi DBCluster SnapshotAttribute

`Memodifikasi DBCluster SnapshotAttribute` menambahkan atribut dan nilai-nilai ke, atau menghapus atribut dan nilai-nilai dari, snapshot klaster DB manual.

Tingkat akses: `Write`.

Jenis sumber daya: cluster-snapshot (wajib).

RDS: memodifikasi DBInstance

`Memodifikasi DBInstance` mengubah pengaturan untuk instans DB.

Tingkat akses: `Write`.

Tindakan bergantung: `iam:PassRole`.

Jenis sumber daya:

- db (wajib).
- pg (wajib).

RDSDBParameter: Modifikasi Grup

`Ubah DBParameter Grup` mengubah parameter dari grup parameter DB.

Tingkat akses: `Write`.

Jenis sumber daya: pg (wajib).

RDSDBSubnet: Modifikasi Grup

Ubah DBSubnet Grup mengubah grup subnet DB yang ada.

Tingkat akses: `Write`.

Jenis sumber daya: subgrp (wajib).

rds: ModifyEventSubscription

ModifyEventSubscription memodifikasi langganan notifikasi peristiwa Neptune yang ada.

Tingkat akses: `Write`.

Jenis sumber daya: es (wajib).

RDS: Reboot DBInstance

Reboot DBInstance memulai ulang layanan mesin basis data untuk instans.

Tingkat akses: `Write`.

Jenis sumber daya: db (wajib).

rds: RemoveRoleFrom DBCluster

RemoveRoleFromDBClustermemisahkan peran AWS Identity and Access Management (IAM) and Access Management (IAM) dari kluster Amazon Neptune DB.

Tingkat akses: `Write`.

Tindakan bergantung: `iam:PassRole`.

Jenis sumber daya: klaster (wajib).

rds: RemoveSourceIdentifierFromSubscription

RemoveSourceIdentifierFromSubscription menghapus pengidentifikasi sumber dari langganan notifikasi peristiwa Neptune yang ada.

Tingkat akses: `Write`.

Jenis sumber daya: es (wajib).

rds: RemoveTagsFromResource

RemoveTagsFromResource menghapus tanda metadata dari sumber daya Neptune.

Tingkat akses: `Tagging`.

Jenis sumber daya:

- cuplikan cluster
- db
- es
- pg
- subgrp

Kunci Kondisi:

- aws:RequestTag/*tag-key*
- aws: TagKeys

RDS: Atur ulang DBCluster ParameterGroup

`Setel ulang DBCluster ParameterGroup` memodifikasi parameter dari grup parameter klaster DB ke nilai default.

Tingkat akses: `Write`.

Jenis sumber daya: cluster-pg (wajib).

RDSDBParameter: Atur Ulang Grup

`Setel Ulang DBParameter Grup`memodifikasi parameter grup parameter DB ke nilai engine/system default.

Tingkat akses: `Write`.

Jenis sumber daya: pg (wajib).

RDS: Kembalikan DBCluster FromSnapshot

`Kembalikan DBCluster FromSnapshot` membuat klaster DB baru dari snapshot klaster DB.

Tingkat akses: `Write`.

Tindakan bergantung: `iam:PassRole`.

Jenis sumber daya:

- cluster (wajib).
- cluster-snapshot (wajib).

Kunci Kondisi:

- aws:RequestTag/*tag-key*
- aws: TagKeys

RDS: Kembalikan DBCluster ToPointInTime

`Kembalikan DBCluster ToPointInTime` memulihkan klaster DB ke titik waktu yang arbitrer.

Tingkat akses: `Write`.

Tindakan bergantung: `iam:PassRole`.

Jenis sumber daya:

- cluster (wajib).
- subgrp (wajib).

Kunci Kondisi:

- aws:RequestTag/*tag-key*
- aws: TagKeys

RDS: Mulai DBCluster

`Mulai DBCluster` memulai klaster DB yang ditentukan.

Tingkat akses: `Write`.

Jenis sumber daya: klaster (wajib).

RDS: berhenti DBCluster

`Berhenti DBCluster` menghentikan klaster DB yang ditentukan.

Tingkat akses: `Write`.

Jenis sumber daya: klaster (wajib).

Jenis sumber daya IAM untuk mengelola Amazon Neptunus

Neptunus mendukung jenis sumber daya dalam tabel berikut untuk digunakan dalam elemen pernyataan kebijakan `Resource` administrasi IAM. Untuk informasi selengkapnya tentang elemen `Resource`, lihat Elemen Kebijakan IAM JSON: Sumber Daya.

Daftar tindakan administrasi Neptunus mengidentifikasi jenis sumber daya yang dapat ditentukan dengan setiap tindakan. Jenis sumber daya juga menentukan kunci kondisi mana yang dapat Anda sertakan dalam kebijakan, sebagaimana ditentukan dalam kolom terakhir tabel di bawah.

`ARN`Kolom dalam tabel di bawah ini menentukan format Amazon Resource Name (ARN) yang harus Anda gunakan untuk referensi sumber daya jenis ini. Bagian yang didahului oleh a `$` harus diganti dengan nilai aktual untuk skenario Anda. Misalnya, jika Anda melihat `$user-name` di ARN, Anda harus mengganti string itu baik dengan nama pengguna IAM yang sebenarnya atau dengan variabel kebijakan yang berisi nama pengguna IAM. Untuk informasi lebih lanjut tentang ARNs, lihat IAM ARNs, danBekerja dengan administrasi ARNs di Amazon Neptunus.

Kolom `Condition Keys` menentukan kunci konteks kondisi yang dapat Anda sertakan dalam pernyataan kebijakan IAM hanya ketika kedua sumber daya ini dan tindakan pendukung yang kompatibel disertakan dalam pernyataan.

Jenis Sumber Daya	ARN	Kunci kondisi
`cluster` (klaster DB)	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :cluster: `*`instance-name`*	aws:ResourceTag/*`tag-key`* rds: tag-klaster/*`tag-key`*
`cluster-pg` (Grup parameter klaster DB)	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :cluster-pg: `*`neptune-DBClusterParameterGroupName`*	aws:ResourceTag/*`tag-key`*

Jenis Sumber Daya	ARN	Kunci kondisi
cluster-snapshot (snapshot klaster DB)	arn:*partition* :rds:*region*:*account-id* :cluster-snapshot: *neptune-DBClusterSnapshotName*	aws:ResourceTag/*tag-key* rds:/cluster-snapshot-tag*tag-key*
db (instans DB)	arn:*partition* :rds:*region*:*account-id* :db:*neptune-DbInstanceName*	aws:ResourceTag/*tag-key* rds: DatabaseClass rds: DatabaseEngine rds:db-tag/ *tag-key*
es (langganan peristiwa)	arn:*partition* :rds:*region*:*account-id* :es:*neptune-CustSubscriptionId*	aws:ResourceTag/*tag-key* rds:es-tag/ *tag-key*
pg (Grup parameter DB)	arn:*partition* :rds:*region*:*account-id* :pg:*neptune-ParameterGroupName*	aws:ResourceTag/*tag-key* rds: pg-tag/ *tag-key*
subgrp (grup subnet DB)	arn:*partition* :rds:*region*:*account-id* :subgrp:*neptune-DBSubnetGroupName* }	aws:ResourceTag/*tag-key* rds:subgrp-tag/ *tag-key*

Kunci kondisi IAM untuk mengelola Amazon Neptunus

Menggunakan kunci kondisi, Anda dapat menentukan kondisi dalam pernyataan kebijakan IAM sehingga pernyataan hanya berlaku ketika kondisi benar. Kunci kondisi yang dapat Anda gunakan dalam pernyataan kebijakan administratif Neptunus termasuk dalam kategori berikut:

- Kunci kondisi global — Ini didefinisikan oleh AWS untuk penggunaan umum dengan AWS layanan. Sebagian besar dapat digunakan dalam pernyataan kebijakan administratif Neptunus.
- Kunci kondisi properti sumber daya administratif — Kunci ini, tercantum di bawah ini, didasarkan pada properti sumber daya administratif.
- Kunci kondisi akses berbasis tag — Kunci ini, tercantum di bawah ini, didasarkan pada AWS tag yang dilampirkan ke sumber daya administratif.

Kunci kondisi properti sumber daya administratif Neptunus

Kunci syarat	Deskripsi	Tipe
`rds:DatabaseClass`	Memfilter akses berdasarkan jenis kelas instans DB.	String
`rds:DatabaseEngine`	Memfilter akses oleh mesin database. Untuk nilai yang mungkin, lihat parameter engine di Create DBInstance API	String
`rds:DatabaseName`	Memfilter akses dengan nama database yang ditentukan pengguna pada instans DB	String
`rds:EndpointType`	Memfilter akses berdasarkan jenis titik akhir. Salah satu dari: PEMBACA, PENULIS, CUSTOM	String
`rds:Vpc`	Memfilter akses berdasarkan nilai yang menentukan apakah instans DB berjalan di Amazon Virtual Private Cloud (Amazon VPC). Untuk menunjukkan bahwa instans DB berjalan di VPC Amazon, tentukan. `true`	Boolean

Kunci kondisi berbasis tag administratif

Amazon Neptune mendukung ketentuan yang menentukan dalam kebijakan IAM menggunakan tanda khusus, untuk mengontrol akses ke Neptune melalui Referensi API Manajemen.

Misalnya, jika Anda menambahkan tanda bernama `environment` ke instans DB Anda, dengan nilai-nilai seperti `beta`, `staging`, dan `production`, Anda kemudian dapat membuat kebijakan yang membatasi akses ke instans berdasarkan nilai tanda tersebut.

Important

Jika Anda mengelola akses ke sumber daya Neptune Anda menggunakan penandaan, pastikan untuk mengamankan akses ke tanda tersebut. Anda dapat membatasi akses untuk tag dengan membuat kebijakan untuk tindakan `AddTagsToResource` dan `RemoveTagsFromResource`.
Misalnya, Anda bisa menggunakan kebijakan berikut untuk menolak daftar tugas pengguna untuk menambah atau menghapus tag semua sumber daya. Anda kemudian dapat membuat kebijakan untuk memungkinkan pengguna tertentu untuk menambahkan atau menghapus tag.

JSON

```
{ "Version":"2012-10-17",
  "Statement":[
    { "Sid": "DenyTagUpdates",
      "Effect": "Deny",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource":"*"
    }
  ]
}
```

Kunci kondisi berbasis tag berikut hanya bekerja dengan sumber daya administratif dalam pernyataan kebijakan administratif.

Kunci kondisi administratif berbasis tag

Kunci syarat	Deskripsi	Tipe
aws:RequestTag/${TagKey}	Memfilter akses berdasarkan keberadaan pasangan nilai kunci tag dalam permintaan.	String

Kunci syarat	Deskripsi	Tipe
aws:ResourceTag/${TagKey}	Memfilter akses berdasarkan pasangan nilai kunci tag yang dilampirkan ke sumber daya.	String
aws:TagKeys	Memfilter akses berdasarkan keberadaan kunci tag dalam permintaan.	String
rds:cluster-pg-tag/${TagKey}	Memfilter akses dengan tag yang dilampirkan ke grup parameter cluster DB.	String
rds:cluster-snapshot-tag/${TagKey}	Memfilter akses dengan tag yang dilampirkan ke snapshot cluster DB.	String
rds:cluster-tag/${TagKey}	Memfilter akses dengan tag yang dilampirkan ke cluster DB.	String
rds:db-tag/${TagKey}	Memfilter akses dengan tag yang dilampirkan ke instance DB.	String
rds:es-tag/${TagKey}	Memfilter akses dengan tag yang dilampirkan ke langganan acara.	String
rds:pg-tag/${TagKey}	Memfilter akses dengan tag yang dilampirkan ke grup parameter DB.	String
rds:req-tag/${TagKey}	Memfilter akses dengan kumpulan kunci tag dan nilai yang dapat digunakan untuk menandai sumber daya.	String
rds:secgrp-tag/${TagKey}	Memfilter akses dengan tag yang dilampirkan ke grup keamanan DB.	String

Kunci syarat	Deskripsi	Tipe
`rds:snapshot-tag/${TagKey}`	Memfilter akses dengan tag yang dilampirkan ke snapshot DB.	String
`rds:subgrp-tag/${TagKey}`	Memfilter akses dengan tag yang dilampirkan ke grup subnet DB	String

Membuat pernyataan kebijakan administratif IAM untuk Amazon Neptunus

Contoh kebijakan administrasi umum

Contoh berikut menunjukkan cara membuat kebijakan administratif Neptunus yang memberikan izin untuk mengambil berbagai tindakan manajemen pada cluster DB.

Kebijakan yang mencegah pengguna IAM menghapus instans DB tertentu

Berikut ini adalah contoh kebijakan yang mencegah pengguna IAM menghapus instance DB Neptunus tertentu:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyDeleteOneInstance",
      "Effect": "Deny",
      "Action": "rds:DeleteDBInstance",
      "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-instance-name"
    }
  ]
}
```

Kebijakan yang memberikan izin untuk membuat instans DB baru

Berikut ini adalah contoh kebijakan yang memungkinkan pengguna IAM untuk membuat instance DB di cluster DB Neptunus tertentu:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateInstance",
      "Effect": "Allow",
      "Action": "rds:CreateDBInstance",
      "Resource": "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster"
    }
  ]
}
```

Kebijakan yang memberikan izin untuk membuat instance DB baru yang menggunakan grup parameter DB tertentu

Berikut ini adalah contoh kebijakan yang memungkinkan pengguna IAM untuk membuat instance DB dalam cluster DB tertentu (di sinius-west-2) di cluster DB Neptunus tertentu hanya menggunakan grup parameter DB tertentu.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateInstanceWithPG",
      "Effect": "Allow",
      "Action": "rds:CreateDBInstance",
      "Resource": [
        "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster",
        "arn:aws:rds:us-west-2:123456789012:pg:my-instance-pg"
      ]
    }
```

```
  ]
}
```

Kebijakan yang memberikan izin untuk mendeskripsikan sumber daya apa pun

Berikut ini adalah contoh kebijakan yang memungkinkan pengguna IAM untuk menggambarkan sumber daya Neptunus apa pun.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDescribe",
            "Effect": "Allow",
            "Action": "rds:Describe*",
            "Resource": "*"
        }
    ]
}
```

Contoh kebijakan administratif berbasis tag

Contoh berikut menunjukkan cara membuat kebijakan administratif Neptunus yang memberi tag untuk memfilter izin untuk berbagai tindakan manajemen pada cluster DB.

Contoh 1: Berikan izin untuk tindakan atas sumber daya menggunakan tanda kustom yang dapat mengambil beberapa nilai

Kebijakan di bawah ini memungkinkan penggunaan `ModifyDBInstance`, `CreateDBInstance` atau API `DeleteDBInstance` pada setiap instans DB yang memiliki tanda `env` diatur ke salah satu dari `dev` atau `test`:

JSON

```
{ "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Sid": "AllowDevTestAccess",
      "Effect": "Allow",
      "Action": [
        "rds:ModifyDBInstance",
        "rds:CreateDBInstance",
        "rds:DeleteDBInstance"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "rds:db-tag/env": [
              "dev",
              "test"
          ],
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Contoh 2: Batasan set kunci dan nilai tag yang dapat digunakan untuk menandai suatu sumber daya

Kebijakan ini menggunakan kunci `Condition` untuk mengizinkan tanda yang memiliki kunci `env` dan nilai `test`, `qa`, atau `dev` yang akan ditambahkan ke sumber daya:

JSON

```
{ "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowTagAccessForDevResources",
      "Effect": "Allow",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "rds:req-tag/env": [
```

```
            "test",
            "qa",
            "dev"
          ],
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Contoh 3: Izinkan akses penuh ke sumber daya Neptunus berdasarkan **`aws:ResourceTag`**

Kebijakan berikut ini mirip dengan contoh pertama di atas, tetapi menggunakan `aws:ResourceTag` sebagai gantinya:

JSON

```
{ "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFullAccessToDev",
      "Effect": "Allow",
      "Action": [
        "rds:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/env": "dev",
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Membuat pernyataan kebijakan IAM khusus untuk mengakses data di Amazon Neptunus

Pernyataan kebijakan akses data Neptunus menggunakan tindakan akses data, sumber daya, dan kunci kondisi, yang semuanya didahului oleh awalan. `neptune-db:`

Topik

- Menggunakan tindakan kueri dalam pernyataan kebijakan akses data Neptunus
- Tindakan IAM untuk akses data di Amazon Neptunus
- Jenis sumber daya IAM untuk mengakses data di Amazon Neptunus
- Kunci kondisi IAM untuk mengakses data di Amazon Neptunus
- Membuat kebijakan akses data IAM di Amazon Neptunus

Menggunakan tindakan kueri dalam pernyataan kebijakan akses data Neptunus

Ada tiga tindakan kueri Neptunus yang dapat digunakan dalam pernyataan kebijakan akses data, yaitu,, dan. `ReadDataViaQuery` `WriteDataViaQuery` `DeleteDataViaQuery` Kueri tertentu mungkin memerlukan izin untuk melakukan lebih dari satu tindakan ini, dan mungkin tidak selalu jelas kombinasi tindakan apa yang harus diizinkan untuk menjalankan kueri.

Sebelum menjalankan kueri, Neptunus menentukan izin yang diperlukan untuk menjalankan setiap langkah kueri, dan menggabungkannya ke dalam set lengkap izin yang dibutuhkan kueri. Perhatikan bahwa set lengkap izin ini mencakup semua tindakan yang mungkin dilakukan kueri, yang belum tentu merupakan kumpulan tindakan yang sebenarnya akan dilakukan kueri saat dijalankan di atas data Anda.

Ini berarti bahwa untuk mengizinkan kueri tertentu dijalankan, Anda harus memberikan izin untuk setiap tindakan yang mungkin dilakukan kueri, apakah itu benar-benar melakukannya atau tidak.

Berikut adalah beberapa contoh pertanyaan Gremlin di mana ini dijelaskan secara lebih rinci:

- ```
  g.V().count()
  ```

  `g.V()`dan `count()` hanya memerlukan akses baca, jadi kueri secara keseluruhan hanya membutuhkan `ReadDataViaQuery` akses.

- ```
  g.addV()
  ```

`addV()`perlu memeriksa apakah simpul dengan ID yang diberikan ada atau tidak sebelum memasukkan yang baru. Ini berarti membutuhkan keduanya `ReadDataViaQuery` dan `WriteDataViaQuery` akses.

- ```
  g.V('1').as('a').out('created').addE('createdBy').to('a')
  ```

  `g.V('1').as('a')`dan `out('created')` hanya memerlukan akses baca, tetapi `addE().from('a')` memerlukan akses baca dan tulis karena `addE()` perlu membaca `from` dan `to` simpul dan memeriksa apakah tepi dengan ID yang sama sudah ada sebelum menambahkan yang baru. Oleh karena itu, kueri secara keseluruhan membutuhkan keduanya `ReadDataViaQuery` dan `WriteDataViaQuery` akses.

- ```
  g.V().drop()
  ```

  `g.V()`hanya membutuhkan akses baca. `drop()`membutuhkan akses baca dan hapus karena perlu membaca simpul atau tepi sebelum menghapusnya, sehingga kueri secara keseluruhan membutuhkan keduanya `ReadDataViaQuery` dan `DeleteDataViaQuery` akses.

- ```
  g.V('1').property(single, 'key1', 'value1')
  ```

  `g.V('1')`hanya membutuhkan akses baca, tetapi `property(single, 'key1', 'value1')` membutuhkan akses baca, tulis, dan hapus. Di sini, `property()` langkah menyisipkan kunci dan nilai jika mereka belum ada di simpul, tetapi jika mereka sudah ada, itu menghapus nilai properti yang ada dan menyisipkan nilai baru di tempatnya. Oleh karena itu, query secara keseluruhan membutuhkan`ReadDataViaQuery`,`WriteDataViaQuery`, dan `DeleteDataViaQuery` akses.

  Setiap kueri yang berisi `property()` langkah akan membutuhkan`ReadDataViaQuery`,`WriteDataViaQuery`, dan `DeleteDataViaQuery` izin.

Berikut adalah beberapa contoh OpenCypher:

- ```
  MATCH (n)
  RETURN n
  ```

  Kueri ini membaca semua node dalam database dan mengembalikannya, yang hanya membutuhkan `ReadDataViaQuery` akses.

- ```
  MATCH (n:Person)
  SET n.dept = 'AWS'
  ```

  Kueri ini membutuhkan`ReadDataViaQuery`,`WriteDataViaQuery`, dan `DeleteDataViaQuery` akses. Ia membaca semua node dengan label 'Orang' dan menambahkan properti baru dengan kunci `dept` dan nilai `AWS` untuk mereka, atau jika `dept` properti sudah ada, itu menghapus nilai lama dan menyisipkan `AWS` sebagai gantinya. Juga, jika nilai yang akan ditetapkan adalah`null`, `SET` menghapus properti sama sekali.

  Karena `SET` klausa mungkin dalam beberapa kasus perlu menghapus nilai yang ada, klausa selalu membutuhkan `DeleteDataViaQuery` izin serta `ReadDataViaQuery` dan `WriteDataViaQuery` izin.

- ```
  MATCH (n:Person)
  DETACH DELETE n
  ```

  Kebutuhan `ReadDataViaQuery` dan `DeleteDataViaQuery` izin kueri ini. Ia menemukan semua node dengan label `Person` dan menghapusnya bersama dengan tepi yang terhubung ke node tersebut dan label dan properti terkait.

- ```
  MERGE (n:Person {name: 'John'})-[:knows]->(:Person {name: 'Peter'})
  RETURN n
  ```

  Kebutuhan `ReadDataViaQuery` dan `WriteDataViaQuery` izin kueri ini. `MERGE`Klausa cocok dengan pola tertentu atau membuatnya. Karena, penulisan dapat terjadi jika pola tidak cocok, izin menulis diperlukan serta izin baca.

Tindakan IAM untuk akses data di Amazon Neptunus

Perhatikan bahwa tindakan akses data Neptunus memiliki awalan, sedangkan tindakan administratif di Neptunus memiliki `neptune-db:` awalan. `rds:`

Nama Sumber Daya Amazon (ARN) untuk sumber daya data di IAM tidak sama dengan ARN yang ditetapkan ke cluster saat pembuatan. Anda harus membangun ARN seperti yang ditunjukkan dalam Menentukan sumber daya data. Sumber daya data tersebut ARNs dapat menggunakan wildcard untuk menyertakan beberapa sumber daya.
  ```

Pernyataan kebijakan akses data juga dapat menyertakan kunci QueryLanguage kondisi neptune-db: untuk membatasi akses menurut bahasa kueri.

Dimulai denganRilis: 1.2.0.0 (2022-07-21), Neptunus mendukung pembatasan izin untuk satu atau lebih tindakan Neptunus tertentu. Ini memberikan kontrol akses yang lebih terperinci daripada yang mungkin sebelumnya.

> **Important**
>
> - Perubahan kebijakan IAM memakan waktu hingga 10 menit untuk diterapkan ke sumber daya Neptune yang ditentukan.
> - Kebijakan IAM yang diterapkan ke klaster DB Neptune berlaku untuk semua contoh dalam klaster itu.

Tindakan akses data berbasis kueri

> **Note**
>
> Tidak selalu jelas izin apa yang diperlukan untuk menjalankan kueri tertentu, karena kueri berpotensi mengambil lebih dari satu tindakan tergantung pada data yang mereka proses. Untuk informasi selengkapnya, lihat Menggunakan tindakan kueri.

`neptune-db:ReadDataViaQuery`

`ReadDataViaQuery`memungkinkan pengguna untuk membaca data dari database Neptunus dengan mengirimkan kueri.

Grup tindakan: baca-saja, baca-tulis.

Kunci konteks tindakan:`neptune-db:QueryLanguage`.

Sumber daya yang dibutuhkan: database.

`neptune-db:WriteDataViaQuery`

`WriteDataViaQuery`memungkinkan pengguna untuk menulis data ke database Neptunus dengan mengirimkan kueri.

Grup aksi: baca-tulis.

Kunci konteks tindakan:`neptune-db:QueryLanguage`.

Sumber daya yang dibutuhkan: database.

`neptune-db:DeleteDataViaQuery`

`DeleteDataViaQuery`memungkinkan pengguna untuk menghapus data dari database Neptunus dengan mengirimkan kueri.

Grup aksi: baca-tulis.

Kunci konteks tindakan:`neptune-db:QueryLanguage`.

Sumber daya yang dibutuhkan: database.

`neptune-db:GetQueryStatus`

`GetQueryStatus`memungkinkan pengguna untuk memeriksa status semua kueri aktif.

Grup tindakan: baca-saja, baca-tulis.

Kunci konteks tindakan:`neptune-db:QueryLanguage`.

Sumber daya yang dibutuhkan: database.

`neptune-db:GetStreamRecords`

`GetStreamRecords`memungkinkan pengguna untuk mengambil catatan aliran dari Neptunus.

Grup aksi: baca-tulis.

Kunci konteks tindakan:`neptune-db:QueryLanguage`.

Sumber daya yang dibutuhkan: database.

`neptune-db:CancelQuery`

`CancelQuery`memungkinkan pengguna untuk membatalkan kueri.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

Tindakan akses data umum

neptune-db:GetEngineStatus

`GetEngineStatus`memungkinkan pengguna untuk memeriksa status mesin Neptunus.

Grup tindakan: baca-saja, baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:GetStatisticsStatus

`GetStatisticsStatus`memungkinkan pengguna untuk memeriksa status statistik yang dikumpulkan untuk database.

Grup tindakan: baca-saja, baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:GetGraphSummary

`GetGraphSummary`API ringkasan grafik memungkinkan Anda untuk mengambil ringkasan hanya-baca dari grafik Anda.

Grup tindakan: baca-saja, baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:ManageStatistics

`ManageStatistics`memungkinkan pengguna untuk mengelola pengumpulan statistik untuk database.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:DeleteStatistics

`DeleteStatistics`memungkinkan pengguna untuk menghapus semua statistik dalam database.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:ResetDatabase

`ResetDatabase`memungkinkan pengguna untuk mendapatkan token yang diperlukan untuk reset dan mengatur ulang database Neptunus.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

Tindakan akses data pemuat massal

neptune-db:StartLoaderJob

`StartLoaderJob`memungkinkan pengguna untuk memulai pekerjaan bulk-loader.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:GetLoaderJobStatus

`GetLoaderJobStatus`memungkinkan pengguna untuk memeriksa status pekerjaan bulk-loader.

Grup tindakan: baca-saja, baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:ListLoaderJobs

`ListLoaderJobs`memungkinkan pengguna untuk membuat daftar semua pekerjaan bulk-loader.

Grup tindakan: hanya daftar, hanya-baca, baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:CancelLoaderJob

`CancelLoaderJob`memungkinkan pengguna untuk membatalkan pekerjaan loader.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

Tindakan akses data pembelajaran mesin

neptune-db:StartMLDataProcessingJob

`StartMLDataProcessingJob`memungkinkan pengguna untuk memulai pekerjaan pemrosesan data Neptunus Neptunus.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:StartMLModelTrainingJob

`StartMLModelTrainingJob`memungkinkan pengguna untuk memulai pekerjaan pelatihan model ML.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:StartMLModelTransformJob

`StartMLModelTransformJob`memungkinkan pengguna untuk memulai pekerjaan transformasi model ML.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:CreateMLEndpoint

`CreateMLEndpoint`memungkinkan pengguna untuk membuat titik akhir Neptunus ML.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:GetMLDataProcessingJobStatus

`GetMLDataProcessingJobStatus`memungkinkan pengguna untuk memeriksa status pekerjaan pemrosesan data Neptunus ML.

Grup tindakan: baca-saja, baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:GetMLModelTrainingJobStatus

`GetMLModelTrainingJobStatus`memungkinkan pengguna untuk memeriksa status pekerjaan pelatihan model Neptunus ML.

Grup tindakan: baca-saja, baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:GetMLModelTransformJobStatus

`GetMLModelTransformJobStatus`memungkinkan pengguna untuk memeriksa status pekerjaan transformasi model Neptunus ML.

Grup tindakan: baca-saja, baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:GetMLEndpointStatus

`GetMLEndpointStatus`memungkinkan pengguna untuk memeriksa status titik akhir Neptunus ML.

Grup tindakan: baca-saja, baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:ListMLDataProcessingJobs

`ListMLDataProcessingJobs`memungkinkan pengguna untuk membuat daftar semua pekerjaan pemrosesan data Neptunus Neptunus.

Grup tindakan: hanya daftar, hanya-baca, baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:ListMLModelTrainingJobs

`ListMLModelTrainingJobs`memungkinkan pengguna untuk membuat daftar semua pekerjaan pelatihan model Neptunus ML.

Grup tindakan: hanya daftar, hanya-baca, baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:ListMLModelTransformJobs

`ListMLModelTransformJobs`memungkinkan pengguna untuk membuat daftar semua pekerjaan transformasi model ML.

Grup tindakan: hanya daftar, hanya-baca, baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:ListMLEndpoints

`ListMLEndpoints`memungkinkan pengguna untuk membuat daftar semua titik akhir Neptunus ML.

Grup tindakan: hanya daftar, hanya-baca, baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:CancelMLDataProcessingJob

`CancelMLDataProcessingJob`memungkinkan pengguna untuk membatalkan pekerjaan pemrosesan data Neptunus Neptunus.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:CancelMLModelTrainingJob

`CancelMLModelTrainingJob`memungkinkan pengguna untuk membatalkan pekerjaan pelatihan model Neptunus ML.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

neptune-db:CancelMLModelTransformJob

`CancelMLModelTransformJob`memungkinkan pengguna untuk membatalkan pekerjaan transformasi model Neptunus ML.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

`neptune-db:DeleteMLEndpoint`

`DeleteMLEndpoint`memungkinkan pengguna untuk menghapus titik akhir Neptunus ML.

Grup aksi: baca-tulis.

Sumber daya yang dibutuhkan: database.

Jenis sumber daya IAM untuk mengakses data di Amazon Neptunus

Sumber daya data, seperti tindakan data, memiliki `neptune-db:` awalan.

Dalam kebijakan akses data Neptunus, Anda menentukan cluster DB yang Anda berikan akses ke dalam ARN dengan format berikut:

```
arn:aws:neptune-db:region:account-id:cluster-resource-id/*
```

Sumber daya seperti ARN berisi bagian-bagian berikut:

- *`region`*adalah AWS Wilayah untuk cluster DB Amazon Neptunus.
- *`account-id`* adalah nomor rekening AWS untuk klaster DB.
- *`cluster-resource-id`* adalah id sumber daya untuk klaster DB.

> **Important**
>
> `cluster-resource-id` berbeda dari pengidentifikasi klaster. Untuk menemukan ID sumber daya cluster di Konsol Manajemen AWS Neptunus, lihat di bagian Konfigurasi untuk cluster DB yang dimaksud.

Kunci kondisi IAM untuk mengakses data di Amazon Neptunus

Menggunakan kunci kondisi, Anda dapat menentukan kondisi dalam pernyataan kebijakan IAM sehingga pernyataan hanya berlaku ketika kondisi benar.

Kunci kondisi yang dapat Anda gunakan dalam pernyataan kebijakan akses data Neptunus termasuk dalam kategori berikut:

- Kunci kondisi global - Subset kunci kondisi AWS global yang didukung Neptunus dalam pernyataan kebijakan akses data tercantum di bawah ini.
- Kunci kondisi khusus layanan — Ini adalah kunci yang ditentukan oleh Neptunus khusus untuk digunakan dalam pernyataan kebijakan akses data. Saat ini hanya ada satu, neptune-db:QueryLanguage, yang memberikan akses hanya jika bahasa kueri tertentu sedang digunakan.

AWS kunci konteks kondisi global yang didukung oleh Neptunus dalam pernyataan kebijakan akses data

Tabel berikut mencantumkan subset kunci konteks kondisi AWS global yang didukung Amazon Neptunus untuk digunakan dalam pernyataan kebijakan akses data:

Kunci kondisi global yang dapat Anda gunakan dalam pernyataan kebijakan akses data

Kunci kondisi	Deskripsi	Tipe
aws:CurrentTime	Memfilter akses berdasarkan tanggal dan waktu permintaan saat ini.	String
aws:EpochTime	Memfilter akses berdasarkan tanggal dan waktu permintaan dinyatakan sebagai nilai epoch UNIX.	Numeric
aws:PrincipalAccount	Memfilter akses oleh akun tempat kepala sekolah yang meminta berada.	String
aws:PrincipalArn	Memfilter akses oleh ARN dari kepala sekolah yang membuat permintaan.	String
aws:PrincipalIsAWSService	Memungkinkan akses hanya jika panggilan dilakukan langsung oleh kepala AWS layanan.	Boolean
aws:PrincipalOrgID	Memfilter akses oleh pengenal AWS organisasi dalam Organizations yang menjadi milik prinsipal yang meminta.	String
aws:PrincipalOrgPaths	Memfilter akses oleh jalur AWS Organizations untuk kepala sekolah yang membuat permintaan.	String

Kunci kondisi	Deskripsi	Tipe
aws:PrincipalTag	Memfilter akses dengan tag yang dilampirkan pada prinsipal yang membuat permintaan.	String
aws:PrincipalType	Memfilter akses berdasarkan jenis prinsipal yang membuat permintaan.	String
aws:RequestedRegion	Memfilter akses oleh AWS Wilayah yang dipanggil dalam permintaan.	String
aws:SecureTransport	Mengizinkan akses hanya jika permintaan dikirim menggunakan SSL.	Boolean
aws:SourceIp	Memfilter akses dengan alamat IP pemohon.	String
aws:TokenIssueTime	Memfilter akses berdasarkan tanggal dan waktu kredensyal keamanan sementara dikeluarkan.	String
aws:UserAgent	Filter mengakses oleh aplikasi klien pemohon.	String
aws:userid	Memfilter akses oleh pengenal utama pemohon.	String
aws:ViaAWSService	Memungkinkan akses hanya jika AWS layanan membuat permintaan atas nama Anda.	Boolean

Kunci kondisi khusus layanan Neptunus

Neptunus mendukung kunci kondisi khusus layanan berikut untuk kebijakan IAM:

Kunci kondisi khusus layanan Neptunus

Kunci kondisi	Deskripsi	Tipe
neptune-db:QueryLanguage	Memfilter akses data dengan bahasa kueri yang digunakan.	String

Kunci kondisi	Deskripsi	Tipe
	Nilai yang valid adalah:`Gremlin`,`OpenCypher` , dan`Sparql`. Tindakan yang didukung adalah`ReadDataViaQuery` ,`WriteDataViaQuery` ,`DeleteDataViaQuery` ,`GetQueryStatus` , dan`CancelQuery` .	

Membuat kebijakan akses data IAM di Amazon Neptunus

Contoh berikut menunjukkan cara membuat kebijakan IAM khusus yang menggunakan kontrol akses berbutir halus bidang data dan APIs tindakan, yang diperkenalkan dalam rilis mesin Neptunus versi 1.2.0.0.

Contoh kebijakan yang memungkinkan akses tidak terbatas ke data dalam cluster DB Neptunus

Contoh kebijakan berikut memungkinkan pengguna IAM untuk terhubung ke cluster DB Neptunus menggunakan otentikasi database IAM, dan menggunakan * "" karakter untuk mencocokkan semua tindakan yang tersedia.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Contoh sebelumnya mencakup ARN sumber daya dalam format yang khusus untuk autentikasi IAM Neptune. Untuk membangun ARN, lihat Menentukan sumber daya data. Perhatikan bahwa ARN yang

digunakan untuk otorisasi `Resource` IAM tidak sama dengan ARN yang ditetapkan ke cluster pada pembuatan.

Contoh kebijakan yang memungkinkan akses hanya-baca ke cluster DB Neptunus

Kebijakan berikut memberikan izin untuk akses hanya-baca penuh ke data dalam klaster DB Neptunus:

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:Read*",
        "neptune-db:Get*",
        "neptune-db:List*"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Contoh kebijakan yang menolak semua akses ke cluster DB Neptunus

Tindakan IAM default adalah menolak akses ke klaster DB kecuali Efek `Allow` diberikan. Namun, kebijakan berikut menolak semua akses ke klaster DB untuk AWS akun dan Wilayah tertentu, yang kemudian diutamakan daripada efek apa pun. `Allow`

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
```

```
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Contoh kebijakan yang memberikan akses baca melalui kueri

Kebijakan berikut hanya memberikan izin untuk membaca dari klaster DB Neptunus menggunakan kueri:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:ReadDataViaQuery",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Contoh kebijakan yang hanya mengizinkan kueri Gremlin

Kebijakan berikut menggunakan kunci `neptune-db:QueryLanguage` kondisi untuk memberikan izin untuk menanyakan Neptunus hanya menggunakan bahasa kueri Gremlin:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "neptune-db:QueryLanguage": "Gremlin"
        }
      }
    }
  ]
}
```

Contoh kebijakan yang memungkinkan semua akses kecuali ke manajemen model Neptunus Neptunus

Kebijakan berikut memberikan akses penuh ke operasi grafik Neptunus kecuali untuk fitur manajemen model Neptunus ML:

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:CancelLoaderJob",
        "neptune-db:CancelQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:DeleteStatistics",
        "neptune-db:GetEngineStatus",
        "neptune-db:GetLoaderJobStatus",
        "neptune-db:GetQueryStatus",
        "neptune-db:GetStatisticsStatus",
        "neptune-db:GetStreamRecords",
        "neptune-db:ListLoaderJobs",
        "neptune-db:ManageStatistics",
        "neptune-db:ReadDataViaQuery",
        "neptune-db:ResetDatabase",
```

```
        "neptune-db:StartLoaderJob",
        "neptune-db:WriteDataViaQuery"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Contoh kebijakan yang memungkinkan akses ke manajemen model Neptunus ML

Kebijakan ini memberikan akses ke fitur manajemen model Neptunus ML:

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:CancelMLDataProcessingJob",
        "neptune-db:CancelMLModelTrainingJob",
        "neptune-db:CancelMLModelTransformJob",
        "neptune-db:CreateMLEndpoint",
        "neptune-db:DeleteMLEndpoint",
        "neptune-db:GetMLDataProcessingJobStatus",
        "neptune-db:GetMLEndpointStatus",
        "neptune-db:GetMLModelTrainingJobStatus",
        "neptune-db:GetMLModelTransformJobStatus",
        "neptune-db:ListMLDataProcessingJobs",
        "neptune-db:ListMLEndpoints",
        "neptune-db:ListMLModelTrainingJobs",
        "neptune-db:ListMLModelTransformJobs",
        "neptune-db:StartMLDataProcessingJob",
        "neptune-db:StartMLModelTrainingJob",
        "neptune-db:StartMLModelTransformJob"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
```

```
}
```

Contoh kebijakan yang memberikan akses kueri penuh

Kebijakan berikut memberikan akses penuh ke operasi kueri grafik Neptunus, tetapi tidak ke fitur seperti reset cepat, aliran, pemuat massal, manajemen model Neptunus, dan sebagainya:

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:GetEngineStatus",
        "neptune-db:GetQueryStatus",
        "neptune-db:CancelQuery"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Contoh kebijakan yang memberikan akses penuh untuk kueri Gremlin saja

Kebijakan berikut memberikan akses penuh ke operasi kueri grafik Neptunus menggunakan bahasa kueri Gremlin, tetapi tidak ke kueri dalam bahasa lain, dan bukan ke fitur seperti reset cepat, aliran, pemuat massal, manajemen model Neptunus, dan sebagainya:

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
```

```
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:GetEngineStatus",
        "neptune-db:GetQueryStatus",
        "neptune-db:CancelQuery"
      ],
      "Resource": [
        "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
      ],
      "Condition": {
        "StringEquals": {
           "neptune-db:QueryLanguage":"Gremlin"
        }
      }
    }
  ]
}
```

Contoh kebijakan yang memberikan akses penuh kecuali untuk reset cepat

Kebijakan berikut memberikan akses penuh ke klaster DB Neptunus kecuali untuk menggunakan reset cepat:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    },
    {
      "Effect": "Deny",
      "Action": "neptune-db:ResetDatabase",
```

```
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Menggunakan peran terkait layanan untuk Amazon Neptunus

Amazon Neptunus AWS Identity and Access Management menggunakan peran terkait layanan (IAM). Peran tertaut layanan adalah tipe IAM role unik yang tertaut langsung ke Neptune. Peran terkait layanan telah ditentukan sebelumnya oleh Neptunus dan mencakup semua izin yang diperlukan layanan untuk memanggil layanan lain atas nama Anda. AWS

> **Important**
>
> Untuk fitur pengelolaan tertentu, Amazon Neptune menggunakan teknologi operasional yang dibagi dengan Amazon RDS. Fitur ini mencakup peran yang terhubung dengan layanan dan izin API manajemen.

Peran terkait layanan memudahkan penggunaan Neptune karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Neptune menentukan izin peran terkait layanan, kecuali jika ditentukan berbeda, hanya Neptune yang dapat mengasumsikan perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran hanya setelah terlebih dahulu menghapus sumber daya terkaitnya. Ini melindungi sumber daya Neptune karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, lihat Layanan AWS yang Berfungsi dengan IAM dan cari layanan yang memiliki Ya di kolom Peran Terkait Layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin Peran yang Terhubung Dengan Layanan untuk Neptune

Neptunus menggunakan peran terkait layanan `AWSServiceRoleForRDS` untuk memungkinkan Neptunus dan Amazon AWS RDS memanggil layanan atas nama instans database Anda. Peran

terkait layanan AWSServiceRoleForRDS memercayai layanan rds.amazonaws.com untuk menjalankan peran.

Kebijakan izin peran memungkinkan Neptune untuk menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

- Tindakan pada ec2:
  - AssignPrivateIpAddresses
  - AuthorizeSecurityGroupIngress
  - CreateNetworkInterface
  - CreateSecurityGroup
  - DeleteNetworkInterface
  - DeleteSecurityGroup
  - DescribeAvailabilityZones
  - DescribeInternetGateways
  - DescribeSecurityGroups
  - DescribeSubnets
  - DescribeVpcAttribute
  - DescribeVpcs
  - ModifyNetworkInterfaceAttribute
  - RevokeSecurityGroupIngress
  - UnassignPrivateIpAddresses
- Tindakan pada sns:
  - ListTopic
  - Publish
- Tindakan pada cloudwatch:
  - PutMetricData
  - GetMetricData
  - CreateLogStream
  - PullLogEvents
  - DescribeLogStreams
  - CreateLogGroup

> **Note**
>
> Anda harus mengonfigurasi izin agar entitas IAM (seperti pengguna, grup, atau peran) dapat membuat, mengedit, atau menghapus peran terkait layanan. Anda mungkin menemukan pesan kesalahan berikut ini:
> Tidak dapat membuat sumber daya. Verifikasi bahwa Anda memiliki izin untuk membuat peran terkait layanan. Jika tidak, tunggu dan coba lagi nanti.
> Jika Anda melihat pesan ini, pastikan bahwa Anda mengaktifkan izin berikut:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
> AWSServiceRoleForRDS",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Untuk informasi selengkapnya, silakan lihat Izin Peran Tertaut Layanan di Panduan Pengguna IAM.

Membuat Peran Terkait Layanan untuk Neptune

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat instans atau klaster, Neptune menciptakan peran terkait layanan untuk Anda.

> **Important**
>
> Untuk informasi lebih lanjut, lihat Peran baru yang muncul di akun IAM saya di Panduan Pengguna IAM.

Jika Anda menghapus peran tertaut layanan ini dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda membuat instans atau klaster Neptune menciptakan peran terkait layanan untuk Anda.

Menyunting Peran Terkait Layanan untuk Neptune

Neptune tidak mengizinkan Anda untuk mengedit peran terkait layanan `AWSServiceRoleForRDS`. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin merujuk peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat Mengedit Peran Tertaut Layanan dalam Panduan Pengguna IAM.

Menghapus Peran Terkait Layanan untuk Neptune

Jika Anda tidak lagi perlu menggunakan fitur atau layanan yang memerlukan peran tertaut layanan, sebaiknya hapus peran tersebut. Dengan begitu, Anda tidak perlu lagi memantau atau memelihara entitas yang tidak digunakan. Namun, Anda harus menghapus semua instans dan kluster Anda sebelum Anda dapat menghapus peran terkait layanan.

Membersihkan Peran Terkait Layanan Sebelum Menghapus

Sebelum dapat menggunakan IAM untuk menghapus peran tertaut layanan, Anda harus mengonfirmasi terlebih dahulu bahwa peran tersebut tidak memiliki sesi aktif dan menghapus sumber daya yang digunakan oleh peran tersebut.

Untuk memeriksa apakah peran terkait layanan memiliki sesi aktif di konsol IAM

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran. Lalu pilih nama (bukan kotak centang) dari peran `AWSServiceRoleForRDS`.
3. Pada halaman Ringkasan untuk peran yang dipilih, pilih tab Penasihat Akses.
4. Di tab Penasihat Akses, tinjau aktivitas terbaru untuk peran terkait layanan tersebut.

> **Note**
>
> Jika Anda tidak yakin apakah Neptune menggunakan peran `AWSServiceRoleForRDS` tersebut, coba hapus peran tersebut. Jika layanan ini menggunakan peran tersebut, peran tidak dapat dihapus dan Anda dapat melihat Wilayah tempat peran tersebut digunakan. Jika peran tersebut sedang digunakan, Anda harus menunggu hingga sesi ini berakhir sebelum dapat menghapus peran tersebut. Anda tidak dapat mencabut sesi untuk peran terkait layanan.

Jika Anda ingin menghapus peran `AWSServiceRoleForRDS` Anda harus terlebih dahulu menghapus semua instans dan kluster Anda.

Menghapus Semua Instans Anda

Gunakan salah satu dari prosedur ini untuk menghapus setiap instans Anda.

Untuk menghapus sebuah instans (konsol)

1. Buka konsol Amazon RDS di https://console.aws.amazon.com/rds/.
2. Di panel navigasi, pilih Instans.
3. Di daftar Instans, pilih instans yang ingin Anda hapus.
4. Pilih Tindakan Instans, lalu pilih Hapus.
5. Jika Anda diminta untuk Buat Snapshot akhir?, pilih Ya atau Tidak.
6. Jika Anda memilih Ya pada langkah sebelumnya, untuk Nama snapshot akhir masukkan nama snapshot akhir Anda.
7. Pilih Hapus.

Untuk menghapus suatu instans (AWS CLI)

Lihat `delete-db-instance` dalam Referensi Perintah AWS CLI .

Untuk menghapus sebuah instans (API)

Lihat `DeleteDBInstance`.

Menghapus Semua Klaster Anda

Gunakan salah satu prosedur berikut untuk menghapus satu kluster, dan kemudian ulangi prosedur untuk setiap kluster Anda.

Untuk menghapus sebuah klaster (konsol)

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di daftar Klaster pilih klaster yang ingin Anda hapus.
3. Pilih Tindakan Klaster, lalu pilih Hapus.
4. Pilih Hapus.

Untuk menghapus sebuah klaster (CLI)

Lihat `delete-db-cluster` dalam Referensi Perintah AWS CLI .

Untuk menghapus sebuah klaster (API)

Lihat `DeleteDBCluster`

Anda dapat menggunakan konsol IAM, IAM CLI, atau IAM API untuk menghapus `AWSServiceRoleForRDS`-peran terkait layanan. Untuk informasi selengkapnya, silakan lihat Menghapus Peran Terkait Layanan di Panduan Pengguna IAM.

Menggunakan kredensi sementara untuk terhubung ke Amazon Neptunus

Amazon Neptune mendukung autentikasi IAM menggunakan kredensials sementara.

Anda dapat menggunakan peran yang diasumsikan untuk mengautentikasi menggunakan kebijakan autentikasi IAM, seperti salah satu contoh kebijakan di bagian sebelumnya.

Jika Anda menggunakan kredensial sementara, Anda harus menentukan `AWS_SESSION_TOKEN` selain `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, dan `SERVICE_REGION`.

> **Note**
>
> Kredensial sementara kedaluwarsa setelah interval tertentu, termasuk token sesinya. Anda harus memperbarui token sesi Anda ketika Anda meminta kredensial baru. Untuk informasi selengkapnya, lihat Menggunakan Kredensil Keamanan Sementara untuk Meminta Akses ke AWS Sumber Daya.

Bagian berikut menjelaskan cara mengizinkan akses dan mengambil kredensial sementara.

Untuk mengautentikasi menggunakan kredensial sementara

1. Buat IAM role dengan izin untuk mengakses kluster Neptune. Untuk informasi lebih lanjut tentang pembuatan peran, lihat the section called “Jenis kebijakan IAM”.
2. Tambahkan hubungan kepercayaan ke peran yang memungkinkan akses ke kredensialnya.

Mengambil kredensial sementara, termasuk `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, dan `AWS_SESSION_TOKEN`.

3. Sambungkan ke klaster Neptune dan tandatangani permintaan menggunakan kredensial sementara. Untuk informasi selengkapnya tentang menyambungkan dann menandatangani permintaan, lihat the section called “Menghubungkan dengan otentikasi IAM”.

Ada berbagai metode untuk mengambil kredensial sementara tergantung pada lingkungan.

Topik

- Mendapatkan Kredensyal Sementara Menggunakan AWS CLI
- Menyiapkan AWS Lambda untuk Otentikasi IAM Neptunus
- Menyiapkan Amazon EC2 untuk Autentikasi IAM Neptune

Mendapatkan Kredensyal Sementara Menggunakan AWS CLI

Untuk mendapatkan kredensil menggunakan AWS Command Line Interface (AWS CLI), pertama-tama Anda perlu menambahkan hubungan kepercayaan yang memberikan izin untuk mengambil peran kepada AWS pengguna yang akan menjalankan perintah. AWS CLI

Tambahkan hubungan kepercayaan berikut ke peran autentikasi IAM Neptune. Jika Anda tidak memiliki peran autentikasi IAM Neptune, lihat the section called “Jenis kebijakan IAM”.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/test"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Untuk informasi tentang menambahkan hubungan kepercayaan ke peran, lihat Mengedit Hubungan Kepercayaan untuk Peran yang Ada dalam Panduan Administrasi AWS Directory Service .

Jika kebijakan Neptune belum melekat pada peran, buat peran baru. Lampirkan kebijakan autentikasi IAM Neptune, dan kemudian tambahkan kebijakan kepercayaan. Untuk informasi selengkapnya tentang membuat peran baru, lihat Membuat Peran Baru.

> **Note**
>
> Bagian berikut mengasumsikan bahwa Anda telah AWS CLI menginstal.

Untuk menjalankan secara AWS CLI manual

1. Masukkan perintah berikut untuk meminta kredensials menggunakan AWS CLI. Ganti ARN peran, nama sesi, dan profil dengan nilai-nilai Anda sendiri.

```
aws sts assume-role  --role-arn  arn:aws:iam::123456789012:role/NeptuneIAMAuthRole
 --role-session-name test --profile testprofile
```

2. Contoh berikut menunjukkan output dari perintah. Bagian `Credentials` berisi nilai-nilai yang Anda butuhkan.

> **Note**
>
> Rekam nilai `Expiration` karena Anda perlu untuk mendapatkan kredensial baru setelah waktu ini.

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:s3-access-example",
        "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-access-
example"
    },
    "Credentials": {
        "SecretAccessKey": "9drTJvcXLB89EXAMPLELB8923FB892xMFI",
        "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/qwjzP2iEXAMPLEbw/
m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
```

```
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8BRi
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=",
        "Expiration": "2016-03-15T00:05:07Z",
        "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
    }
}
```

3. Atur variabel lingkungan menggunakan kredensial yang dikembalikan.

```
export AWS_ACCESS_KEY_ID=ASIAJEXAMPLEXEG2JICEA
export AWS_SECRET_ACCESS_KEY=9drTJvcXLB89EXAMPLELB8923FB892xMFI
export AWS_SESSION_TOKEN=AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/qwjzP2iEXAMPLEbw/
m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8BRi
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=

export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      ca-west-1 or sa-east-1 or eu-north-1 or eu-south-2 or eu-
west-1 or eu-west-2 or eu-west-3 or
                      eu-central-1 or me-south-1 or me-central-1 or il-central-1 or
 af-south-1 or ap-east-1 or
                      ap-northeast-1 or ap-northeast-2 or ap-northeast-3 or ap-
southeast-1 or ap-southeast-2 or
                      ap-southeast-3 or ap-southeast-4 or ap-southeast-5 or ap-
south-1 or ap-south-2 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

4. Sambungkan menggunakan salah satu metode berikut.

   - the section called “Menghubungkan dari Gremlin Console”
   - the section called “Menghubungkan menggunakan Gremlin Java”
   - the section called “Terhubung dengan SPARQL Java”
   - the section called “Menghubungkan dengan Python”

Untuk menggunakan skrip untuk mendapatkan kredensialnya

1. Jalankan perintah berikut untuk menginstal perintah jq. Script menggunakan perintah ini untuk mengurai output dari AWS CLI perintah.

```
sudo yum -y install jq
```

2. Buat file bernama `credentials.sh` di editor teks dan tambahkan teks berikut. Ganti Wilayah layanan, ARN peran, nama sesi, dan profil dengan nilai-nilai Anda sendiri.

```
#!/bin/bash

creds_json=$(aws sts assume-role  --role-arn  arn:aws:iam::123456789012:role/
NeptuneIAMAuthRole --role-session-name test --profile testprofile)

export AWS_ACCESS_KEY_ID=$(echo "$creds_json" | jq .Credentials.AccessKeyId |tr -d
 '"')
export AWS_SECRET_ACCESS_KEY=$(echo "$creds_json" |
 jq .Credentials.SecretAccessKey| tr -d '"')
export AWS_SESSION_TOKEN=$(echo "$creds_json" | jq .Credentials.SessionToken|tr -d
 '"')

export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      ca-west-1 or sa-east-1 or eu-north-1 or eu-south-2 or eu-
west-1 or eu-west-2 or eu-west-3 or
                      eu-central-1 or me-south-1 or me-central-1 or il-central-1 or
 af-south-1 or ap-east-1 or
                      ap-northeast-1 or ap-northeast-2 or ap-northeast-3 or ap-
southeast-1 or ap-southeast-2 or
                      ap-southeast-3 or ap-southeast-4 or ap-southeast-5 or ap-
south-1 or ap-south-2 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

3. Sambungkan menggunakan salah satu metode berikut.

   - the section called “Menghubungkan dari Gremlin Console”
   - the section called “Menghubungkan menggunakan Gremlin Java”
   - the section called “Terhubung dengan SPARQL Java”
   - the section called “Menghubungkan dengan Python”

Menyiapkan AWS Lambda untuk Otentikasi IAM Neptunus

AWS Lambda menyertakan kredensyal secara otomatis setiap kali fungsi Lambda dijalankan.

Pertama Anda menambahkan hubungan kepercayaan yang memberikan izin untuk mengambil peran untuk layanan Lambda.

Tambahkan hubungan kepercayaan berikut ke peran autentikasi IAM Neptune. Jika Anda tidak memiliki peran autentikasi IAM Neptune, lihat the section called “Jenis kebijakan IAM”.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Untuk informasi tentang menambahkan hubungan kepercayaan ke peran, lihat Mengedit Hubungan Kepercayaan untuk Peran yang Ada dalam Panduan Administrasi AWS Directory Service.

Jika kebijakan Neptune belum melekat pada peran, buat peran baru. Lampirkan kebijakan autentikasi IAM Neptune, dan kemudian tambahkan kebijakan kepercayaan. Untuk informasi tentang pembuatan peran baru, lihat Membuat Peran Baru dalam Panduan Administrasi AWS Directory Service .

Untuk mengakses Neptune dari Lambda

1. Masuk ke Konsol Manajemen AWS dan buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Buat fungsi Lambda baru untuk Python versi 3.6.

3. Tetapkan peran `AWSLambdaVPCAccessExecutionRole` ke fungsi Lambda. Ini diperlukan untuk mengakses sumber daya Neptune, yang hanya berupa VPC.
4. Tetapkan IAM role autentikasi Neptune ke fungsi Lambda.

   Untuk informasi lebih lanjut, lihat Izin Lambda AWS dalam Panduan Developer AWS Lambda .
5. Salin sampel Python autentikasi IAM ke dalam kode fungsi Lambda.

   Untuk informasi selengkapnya tentang sampel dan kode sampel, lihat the section called “Menghubungkan dengan Python”.

Menyiapkan Amazon EC2 untuk Autentikasi IAM Neptune

Amazon EC2 memungkinkan Anda untuk menggunakan profil instans untuk secara otomatis memberikan kredensial. Untuk informasi lebih lanjut, lihat Menggunakan Profil Instans dalam Panduan Pengguna IAM.

Pertama, Anda menambahkan hubungan kepercayaan yang memberikan izin untuk mengambil peran ke layanan Amazon EC2.

Tambahkan hubungan kepercayaan berikut ke peran autentikasi IAM Neptune. Jika Anda tidak memiliki peran autentikasi IAM Neptune, lihat the section called “Jenis kebijakan IAM”.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Untuk informasi tentang menambahkan hubungan kepercayaan ke peran, lihat Mengedit Hubungan Kepercayaan untuk Peran yang Ada dalam Panduan Administrasi AWS Directory Service .

Jika kebijakan Neptune belum melekat pada peran, buat peran baru. Lampirkan kebijakan autentikasi IAM Neptune, dan kemudian tambahkan kebijakan kepercayaan. Untuk informasi tentang pembuatan peran baru, lihat Membuat Peran Baru dalam Panduan Administrasi AWS Directory Service .

Untuk menggunakan skrip untuk mendapatkan kredensialnya

1. Jalankan perintah berikut untuk menginstal perintah jq. Script menggunakan perintah ini untuk mengurai output dari perintah curl.

```
sudo yum -y install jq
```

2. Buat file bernama `credentials.sh` di editor teks dan tambahkan teks berikut. Ganti Wilayah layanan dengan nilai Anda sendiri.

```
TOKEN=$( curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600" )
role_name=$( curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/
latest/meta-data/iam/security-credentials/ )
creds_json=$( curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/
latest/meta-data/iam/security-credentials/${role_name} )

export AWS_ACCESS_KEY_ID=$(echo "$creds_json" | jq .AccessKeyId |tr -d '"')
export AWS_SECRET_ACCESS_KEY=$(echo "$creds_json" | jq .SecretAccessKey| tr -d '"')
export AWS_SESSION_TOKEN=$(echo "$creds_json" | jq .Token|tr -d '"')

export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      ca-west-1 or sa-east-1 or eu-north-1 or eu-south-2 or eu-
west-1 or eu-west-2 or eu-west-3 or
                      eu-central-1 or me-south-1 or me-central-1 or il-central-1 or
 af-south-1 or ap-east-1 or
                      ap-northeast-1 or ap-northeast-2 or ap-northeast-3 or ap-
southeast-1 or ap-southeast-2 or
                      ap-southeast-3 or ap-southeast-4 or ap-southeast-5 or ap-
south-1 or ap-south-2 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

3. Jalankan skrip di shell `bash` menggunakan perintah `source`:

```
source credentials.sh
```

Lebih baik lagi adalah menambahkan perintah dalam script ini ke file `.bashrc` pada instans EC2 Anda sehingga mereka akan dipanggil secara otomatis ketika Anda login, membuat kredensial sementara tersedia untuk konsol Gremlin.

4. Sambungkan menggunakan salah satu metode berikut.

   - the section called “Menghubungkan dari Gremlin Console”
   - the section called “Menghubungkan menggunakan Gremlin Java”
   - the section called “Terhubung dengan SPARQL Java”
   - the section called “Menghubungkan dengan Python”

Pencatatan dan pemantauan penggunaan dan kinerja di Amazon Neptunus

Amazon Neptune mendukung berbagai metode untuk memantau kinerja dan penggunaan:

- Status klaster— Periksa kesehatan mesin database grafik klaster Neptune. Untuk informasi selengkapnya, lihat the section called “Status instans”.
- Amazon CloudWatch — Neptunus secara otomatis mengirim metrik CloudWatch ke dan juga mendukung Alarm. CloudWatch Untuk informasi selengkapnya, lihat the section called “Menggunakan CloudWatch”.
- File log audit— Lihat, unduh, atau tonton file log basis data menggunakan konsol Neptune. Untuk informasi selengkapnya, lihat the section called “Log Audit dengan Neptune”.
- Menerbitkan log ke Amazon CloudWatch Logs — Anda dapat mengonfigurasi cluster DB Neptunus untuk mempublikasikan data log audit ke grup log di Amazon Logs. CloudWatch Dengan CloudWatch Log, Anda dapat melakukan analisis real-time dari data log, digunakan CloudWatch untuk membuat alarm dan melihat metrik, dan menggunakan CloudWatch Log untuk menyimpan catatan log Anda dalam penyimpanan yang sangat tahan lama. Untuk informasi selengkapnya, lihat Log Neptunus CloudWatch .
- AWS CloudTrail- Neptunus mendukung pencatatan API menggunakan. CloudTrail Untuk informasi selengkapnya, lihat the section called “Mencatat Panggilan API Neptunus dengan AWS CloudTrail”.

- Penandaan— Gunakan tanda untuk menambahkan metadata ke sumber daya Neptune Anda dan lacak penggunaan berdasarkan tanda. Untuk informasi selengkapnya, lihat the section called “Menandai sumber daya Neptunus”.

Amazon Neptunus dan titik akhir VPC antarmuka ()AWS PrivateLink

Anda dapat menggunakan AWS PrivateLinkuntuk membuat koneksi pribadi antara VPC dan titik akhir API Amazon Neptunus. Anda dapat mengakses operasi API Amazon Neptunus seolah-olah ada di VPC Anda, tanpa menggunakan gateway internet, perangkat NAT, koneksi VPN, atau koneksi. Lalu lintas antara VPC Anda dan Amazon Neptunus tetap berada dalam jaringan Amazon.

Amazon Neptune membagikan infrastruktur API dengan Amazon Relational Database Service. Untuk menyiapkan titik akhir VPC antarmuka untuk Amazon Neptunus, ikuti petunjuk di Amazon RDS API dan antarmuka VPC endpoint () di Panduan Pengguna Amazon Aurora.AWS PrivateLink

Pertimbangan kepatuhan untuk Amazon Neptunus

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya tentang tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS, lihat Dokumentasi AWS Keamanan.

Membangun penyebaran Amazon Neptunus yang tangguh dan toleran terhadap bencana

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona

Ketersediaan, Anda dapat merancang dan mengoperasikan aplikasi dan basis data yang secara otomatis melakukan failover di antara Zona Ketersediaan tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur biasa yang terdiri dari satu atau beberapa pusat data.

Sebuah klaster DB Amazon Neptune hanya dapat dibuat dalam Amazon VPC yang memiliki setidaknya dua subnet di setidaknya dua Availability Zone. Dengan mendistribusikan instans klaster di setidaknya dua Availability Zone, Neptune membantu memastikan bahwa ada instans yang tersedia di klaster Anda apabila terjadi kegagalan Availability Zone yang tidak dimungkinkan. Volume klaster untuk klaster DB Neptune Anda selalu mencakup tiga Availability Zone untuk menyediakan penyimpanan yang tahan lama dengan kemungkinan kehilangan data yang lebih sedikit.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat Infrastruktur AWS Global.

Migrasi grafik yang ada ke Amazon Neptune

Ada sejumlah alat dan teknik yang dapat membantu Anda memigrasi data grafik yang ada ke Amazon Neptune dari penyimpanan data lain.

Alur kerja migrasi sederhana melibatkan langkah-langkah berikut:

- Mengekspor data dari penyimpanan yang ada ke Amazon Simple Storage Service (Amazon S3).
- Bersihkan dan format data untuk diimpor.
- Memuatnya ke dalam klaster DB Neptune menggunakan Bulk Loader Neptune.
- Mengkonfigurasi aplikasi Gremlin atau SPARQL Anda untuk menggunakan ttik akhir yang sesuai dengan yang disediakan Neptune .

> **Note**
>
> Klaster Neptune Anda harus berjalan di VPC yang dapat diakses oleh aplikasi Anda.

Ada cara untuk menyederhanakan dan mengotomatisasi beberapa langkah berikut, tergantung di mana data disimpan:

Topik

- Migrasi dari Neo4j ke Amazon Neptune
- Migrasi grafik yang ada dari server Apache TinkerPop Gremlin ke Amazon Neptunus
- Migrasi grafik yang ada dari toko triple RDF ke Amazon Neptune
- Menggunakan AWS Database Migration Service (AWS DMS) untuk bermigrasi dari database relasional atau NoSQL ke Amazon Neptunus
- Migrasi dari Blazegraph ke Amazon Neptune

Migrasi dari Neo4j ke Amazon Neptune

Neo4j dan Amazon Neptunus keduanya adalah database grafik, dirancang untuk beban kerja grafik transaksional online yang mendukung model data grafik properti berlabel. Kesamaan ini membuat Neptunus menjadi pilihan umum bagi pelanggan yang ingin memigrasikan aplikasi Neo4j mereka saat ini. Namun, migrasi ini tidak hanya lift dan shift, karena ada perbedaan dalam bahasa dan dukungan fitur, karakteristik operasional, arsitektur server, dan kemampuan penyimpanan antara dua database.

Halaman ini membingkai proses migrasi dan memunculkan hal-hal yang perlu dipertimbangkan sebelum memigrasikan aplikasi grafik Neo4j ke Neptunus. Pertimbangan ini berlaku secara umum untuk aplikasi grafik Neo4j apa pun, baik yang didukung oleh basis data Komunitas, Perusahaan, atau Aura. Meskipun setiap solusi unik dan mungkin memerlukan prosedur tambahan, semua migrasi mengikuti pola umum yang sama.

Setiap langkah yang dijelaskan dalam bagian berikut mencakup pertimbangan dan rekomendasi untuk menyederhanakan proses migrasi. Selain itu, ada alat sumber terbuka, dan posting blog yang menggambarkan proses, dan bagian kompatibilitas fitur dengan opsi arsitektur yang direkomendasikan.

Topik

- Informasi umum tentang migrasi dari Neo4j ke Neptunus
- Bersiap untuk bermigrasi dari Neo4j ke Neptunus
- Penyediaan infrastruktur saat bermigrasi dari Neo4j ke Neptunus
- Migrasi data dari Neo4j ke Neptunus
- Migrasi aplikasi dari Neo4j ke Neptunus
- Kompatibilitas Neptunus dengan Neo4j
- Menulis ulang kueri Cypher untuk dijalankan di OpenCypher di Neptunus
- Sumber daya untuk bermigrasi dari Neo4j ke Neptunus

Informasi umum tentang migrasi dari Neo4j ke Neptunus

Dengan dukungan Neptunus untuk bahasa kueri OpenCypher, Anda dapat memindahkan sebagian besar beban kerja Neo4j yang menggunakan protokol Bolt atau HTTPS ke Neptunus. Namun, OpenCypher adalah spesifikasi open-source yang berisi sebagian besar tetapi tidak semua fungsi yang didukung oleh database lain seperti Neo4j.

Meskipun kompatibel dalam banyak hal, Neptunus bukanlah pengganti drop-in untuk Neo4j. Neptunus adalah layanan database grafik yang dikelola sepenuhnya dengan fitur perusahaan seperti ketersediaan tinggi dan daya tahan tinggi yang secara arsitektur berbeda dari Neo4j. Neptunus berbasis instans, dengan satu instance penulis utama dan hingga 15 contoh replika baca yang memungkinkan Anda menskalakan kapasitas baca secara horizontal. Menggunakan Neptunus Tanpa Server, Anda dapat secara otomatis menskalakan kapasitas komputasi Anda ke atas dan ke bawah tergantung pada volume kueri. Ini tidak tergantung pada penyimpanan Neptunus, yang diskalakan secara otomatis saat Anda menambahkan data.

Neptunus mendukung spesifikasi standar OpenCypher open-source, versi 9. Di AWS, kami percaya bahwa open source baik untuk semua orang dan kami berkomitmen untuk membawa nilai open source kepada pelanggan kami, dan untuk membawa keunggulan operasional AWS ke komunitas open source.

Namun, banyak aplikasi yang berjalan di Neo4j juga menggunakan fitur berpemilik yang tidak bersumber terbuka dan Neptunus tidak mendukung. Misalnya, Neptunus tidak mendukung prosedur APOC, beberapa klausa dan fungsi khusus Cypher, dan,, atau tipe data. `Char` `Date` `Duration` Neptunus melakukan transmisi otomatis tipe data yang hilang ke tipe data yang didukung.

Selain OpenCypher, Neptunus juga mendukung bahasa kueri Apache TinkerPop Gremlin untuk grafik properti (serta SPARQL untuk data RDF). Gremlin dapat beroperasi dengan OpenCypher pada grafik properti yang sama, dan dalam banyak kasus Anda dapat menggunakan Gremlin untuk menyediakan fungsionalitas yang tidak disediakan OpenCypher. Di bawah ini adalah perbandingan singkat dari kedua bahasa tersebut:

	OpenCypher	Gremlin
Gaya	Deklaratif	Imperatif

Sintaksis	Pencocokan pola
	`Match p=(a)-[:route]->(d)`
	`WHERE a.code='ANC'`
	`RETURN p`
		Berbasis traversal
		`g.V().has('code', 'ANC').`
		`out('route').path().`
		`by(elementMap())`

	OpenCypher	Gremlin
Kemudahan penggunaan	Terinspirasi SQL, dapat dibaca oleh non-programmer	Kurva belajar yang lebih curam, mirip dengan bahasa pemrograman seperti Java
Fleksibilitas	Rendah	Tinggi
Dukungan kueri	Kueri berbasis string	Kueri berbasis string atau kode in-line yang didukung oleh pustaka klien
Klien	HTTPS dan Bolt	HTTPS dan Websockets

Secara umum, tidak perlu mengubah model data Anda untuk bermigrasi dari Neo4j ke Neptunus, karena Neo4j dan Neptunus mendukung data grafik properti berlabel (LPG). Namun, Neptunus memiliki beberapa perbedaan arsitektur dan model data yang dapat Anda manfaatkan untuk mengoptimalkan kinerja. Contoh:

- IDs Neptunus diperlakukan sebagai warga negara kelas satu.
- Neptunus AWS Identity and Access Management menggunakan kebijakan (IAM) untuk mengamankan akses ke data grafik Anda dengan cara yang fleksibel dan terperinci.
- Neptunus menyediakan beberapa cara untuk menggunakan notebook Jupyter untuk menjalankan kueri dan memvisualisasikan hasilnya. Neptunus juga bekerja dengan alat visualisasi pihak ketiga.
- > Meskipun Neptunus tidak memiliki pengganti drop-in untuk perpustakaan Neo4j Graph Data Science (GDS), Neptunus mendukung analisis grafik saat ini melalui berbagai solusi. Misalnya, beberapa contoh notebook menunjukkan cara memanfaatkan integrasi Neptunus dengan SDK AWS Pandas dalam lingkungan Python untuk menjalankan analitik pada data grafik.

Silakan hubungi untuk AWS mendukung atau melibatkan tim AWS akun Anda jika Anda memiliki pertanyaan. Kami menggunakan umpan balik Anda untuk memprioritaskan fitur baru yang akan memenuhi kebutuhan Anda.

Bersiap untuk bermigrasi dari Neo4j ke Neptunus

Migrasi dari database grafik Neo4j ke layanan database grafik Neptunus dapat didekati dengan salah satu dari dua cara utama: re-platforming atau refactoring/re-architecting. Pendekatan re-platforming melibatkan modifikasi model data dan arsitektur aplikasi yang ada untuk memanfaatkan kemampuan Neptunus dengan sebaik-baiknya, sedangkan pendekatan refactoring berfokus pada menemukan komponen yang setara di Neptunus untuk membangun implementasi yang sebanding. Dalam praktiknya, kombinasi dari strategi ini sering digunakan, karena proses migrasi melibatkan penyeimbangan arsitektur target Neptunus dengan kendala dan persyaratan dari implementasi Neo4j yang ada. Terlepas dari pendekatannya, kuncinya adalah bekerja mundur dari kasus penggunaan aplikasi untuk merancang model data, kueri, dan arsitektur keseluruhan yang paling sesuai dengan kebutuhan Anda.

Pendekatan untuk bermigrasi

Saat memigrasikan aplikasi Neo4j ke Neptunus, kami merekomendasikan salah satu dari dua strategi: re-platforming, atau refactoring/re-architecting. Untuk informasi selengkapnya tentang strategi migrasi, lihat 6 Strategi untuk Migrasi Aplikasi ke Cloud, posting blog oleh Stephen Orban.

Pendekatan re-platforming, kadang-kadang disebut lift-tinker-and-shift, melibatkan langkah-langkah berikut:

- Identifikasi kasus penggunaan yang dimaksudkan untuk dipenuhi oleh aplikasi Anda.
- Ubah model data grafik dan arsitektur aplikasi yang ada untuk mengatasi kebutuhan beban kerja ini dengan sebaik-baiknya menggunakan kemampuan Neptunus.
- Tentukan cara memigrasikan data, kueri, dan bagian lain dari aplikasi sumber ke dalam model dan arsitektur target.

Pendekatan yang bekerja mundur ini memungkinkan Anda memigrasikan aplikasi Anda ke jenis solusi Neptunus yang dapat Anda rancang jika ini adalah proyek baru.

Sebaliknya, pendekatan refactoring melibatkan:

- Mengidentifikasi komponen implementasi yang ada, termasuk infrastruktur, data, kueri, dan kemampuan aplikasi.
- Menemukan ekuivalen di Neptunus yang dapat digunakan untuk membangun implementasi yang sebanding.

Pendekatan working-forward ini berusaha untuk menukar satu implementasi dengan implementasi lainnya.

Dalam praktiknya, Anda cenderung mengadopsi campuran dari dua pendekatan ini. Anda mungkin mulai dengan kasus penggunaan, merancang arsitektur Neptunus target, tetapi kemudian beralih ke implementasi Neo4j yang ada untuk mengidentifikasi batasan dan invarian yang harus Anda pertahankan. Misalnya, Anda mungkin harus terus berintegrasi dengan sistem eksternal lainnya, atau terus menawarkan khusus APIs kepada konsumen aplikasi grafik Anda. Dengan informasi ini, Anda dapat menentukan data apa yang sudah ada untuk pindah ke model target Anda, dan apa yang harus bersumber di tempat lain.

Di titik lain, Anda mungkin mulai dengan menganalisis bagian tertentu dari implementasi Neo4j Anda sebagai sumber informasi terbaik tentang pekerjaan yang dimaksudkan untuk dilakukan oleh aplikasi Anda. Arkeologi semacam itu dalam aplikasi yang ada dapat membantu menentukan kasus penggunaan yang kemudian dapat Anda rancang untuk menggunakan kemampuan Neptunus.

Baik Anda sedang membangun aplikasi baru menggunakan Neptunus atau memigrasikan aplikasi yang sudah ada dari Neo4j, sebaiknya Anda bekerja mundur dari kasus penggunaan untuk merancang model data, serangkaian kueri, dan arsitektur aplikasi yang memenuhi kebutuhan bisnis Anda.

Perbedaan arsitektur antara Neptunus dan Neo4j

Ketika pelanggan pertama kali mempertimbangkan untuk memigrasikan aplikasi dari Neo4j ke Neptunus, seringkali tergoda untuk melakukan perbandingan berdasarkan ukuran instans. like-to-like Namun, arsitektur Neo4j dan Neptunus memiliki perbedaan mendasar. Neo4j didasarkan pada all-in-one pendekatan di mana pemuatan data, ETL data, kueri aplikasi, penyimpanan data, dan operasi manajemen semuanya terjadi dalam kumpulan sumber daya komputasi yang sama, seperti instans EC2.

Neptunus, sebaliknya, adalah database grafik terfokus OLTP di mana arsitektur memisahkan tanggung jawab dan di mana sumber daya dipisahkan sehingga mereka dapat skala secara dinamis dan independen.

Saat bermigrasi dari Neo4j ke Neptunus, tentukan persyaratan ketahanan, ketersediaan, dan skalabilitas data aplikasi Anda. Arsitektur cluster Neptunus menyederhanakan desain aplikasi yang membutuhkan daya tahan tinggi, ketersediaan dan skalabilitas. Dengan pemahaman tentang arsitektur cluster Neptunus, Anda kemudian dapat merancang topologi cluster Neptunus untuk memenuhi persyaratan ini.

Arsitektur cluster Neo4j

Banyak aplikasi produksi menggunakan pengelompokan kausal Neo4j untuk memberikan daya tahan data, ketersediaan tinggi, dan skalabilitas. Arsitektur pengelompokan Neo4j menggunakan instance core-server dan read-replica:

- Server inti menyediakan daya tahan data dan toleransi kesalahan dengan mereplikasi data menggunakan protokol Raft.
- Replika baca menggunakan pengiriman log transaksi untuk mereplikasi data secara asinkron untuk beban kerja throughput baca yang tinggi.

Setiap instance dalam cluster, apakah server inti atau replika baca, berisi salinan lengkap data grafik.

Arsitektur cluster Neptunus

Cluster Neptunus terdiri dari contoh penulis utama dan hingga 15 contoh replika baca. Semua instance di cluster berbagi layanan penyimpanan terdistribusi dasar yang sama yang terpisah dari instance.

- Instance penulis utama mengoordinasikan semua operasi penulisan ke database dan dapat diskalakan secara vertikal untuk memberikan dukungan fleksibel untuk beban kerja penulisan yang berbeda. Ini juga mendukung operasi baca.
- Instans replika baca mendukung operasi baca dari volume penyimpanan yang mendasarinya, dan memungkinkan Anda menskalakan secara horizontal untuk mendukung beban kerja baca tinggi. Mereka juga menyediakan ketersediaan tinggi dengan berfungsi sebagai target failover untuk instance utama.

  > **Note**
  >
  > Untuk beban kerja tulis yang berat, yang terbaik adalah menskalakan instance replika baca ke ukuran yang sama dengan instance penulis, untuk memastikan bahwa pembaca dapat tetap konsisten dengan perubahan data.

- Volume penyimpanan yang mendasari menskalakan kapasitas penyimpanan secara otomatis saat data dalam database Anda meningkat, hingga 128 tebibytes (TiB) penyimpanan.

Ukuran instans bersifat dinamis dan independen. Setiap instance dapat diubah ukurannya saat cluster berjalan, dan replika baca dapat ditambahkan atau dihapus saat cluster sedang berjalan.

Fitur Neptunus Tanpa Server dapat meningkatkan kapasitas komputasi Anda naik dan turun secara otomatis saat permintaan naik dan turun. Hal ini tidak hanya dapat mengurangi overhead administratif Anda, ini juga memungkinkan Anda mengonfigurasi database untuk menangani lonjakan besar dalam permintaan tanpa menurunkan kinerja atau mengharuskan Anda untuk penyediaan berlebihan.

Anda dapat menghentikan cluster Neptunus hingga 7 hari.

Neptunus juga mendukung auto-scaling, untuk menyesuaikan ukuran instans pembaca secara otomatis berdasarkan beban kerja.

Menggunakan fitur database global Neptunus, Anda dapat mencerminkan cluster di hingga 5 wilayah lain.

Neptunus juga toleran terhadap kesalahan menurut desain:

- Volume cluster yang menyediakan penyimpanan data ke semua instance di cluster mencakup beberapa Availability Zones (AZs) dalam satu. Wilayah AWS Setiap AZ berisi salinan lengkap data cluster.
- Jika instance utama menjadi tidak tersedia, Neptunus secara otomatis gagal ke replika baca yang ada dengan nol kehilangan data, biasanya dalam waktu kurang dari 30 detik. Jika tidak ada replika baca yang ada di cluster, Neptunus secara otomatis menyediakan contoh utama baru — sekali lagi, dengan nol kehilangan data.

Apa artinya semua ini adalah bahwa ketika bermigrasi dari cluster kausal Neo4j ke Neptunus, Anda tidak perlu merancang topologi cluster secara eksplisit untuk daya tahan data yang tinggi dan ketersediaan tinggi. Ini membuat Anda mengukur cluster Anda untuk beban kerja baca dan tulis yang diharapkan, dan persyaratan ketersediaan yang meningkat yang mungkin Anda miliki, hanya dalam beberapa cara:

- Untuk menskalakan operasi baca, tambahkan instance replika baca atau aktifkan fungsionalitas Neptunus Tanpa Server.
- Untuk meningkatkan ketersediaan, distribusikan instance utama dan baca replika di klaster Anda melalui beberapa Availability Zones (AZs).
- Untuk mengurangi waktu failover, sediakan setidaknya satu instance replika baca yang dapat berfungsi sebagai target failover untuk primer. Anda dapat menentukan urutan di mana instance replika baca dipromosikan ke primer setelah kegagalan dengan menetapkan prioritas setiap replika. Ini adalah praktik terbaik untuk memastikan bahwa target failover memiliki kelas instance yang mampu menangani beban kerja tulis aplikasi Anda jika dipromosikan ke primer.

Perbedaan penyimpanan data antara Neptunus dan Neo4j

Neptunus menggunakan model data grafik berdasarkan model quad asli. Saat memigrasikan data Anda ke Neptunus, ada beberapa perbedaan dalam arsitektur model data dan lapisan penyimpanan yang harus Anda ketahui untuk memanfaatkan penyimpanan bersama terdistribusi dan terukur yang disediakan Neptunus secara optimal:

- Neptunus tidak menggunakan skema atau batasan yang didefinisikan secara eksplisit. Ini memungkinkan Anda menambahkan node, tepi, dan properti secara dinamis tanpa harus menentukan skema sebelumnya. Neptunus tidak membatasi nilai dan jenis data yang disimpan, kecuali seperti yang dicatat dalam batas Neptunus. Sebagai bagian dari arsitektur penyimpanan Neptunus, data juga secara otomatis diindeks dengan cara yang menangani banyak pola akses yang paling umum. Arsitektur penyimpanan ini menghilangkan overhead operasional pembuatan dan pengelolaan skema database dan optimasi indeks.
- Neptunus menyediakan arsitektur penyimpanan terdistribusi dan bersama yang unik yang secara otomatis menskalakan dalam potongan 10 GB seiring bertambahnya kebutuhan penyimpanan database Anda, hingga 128 tebibytes (TiB). Lapisan penyimpanan ini andal, tahan lama, dan toleran terhadap kesalahan, dengan data disalin 6 kali, dua kali di masing-masing 3 Availability Zone. Ini menyediakan semua cluster Neptunus dengan lapisan penyimpanan data yang sangat tersedia dan toleran kesalahan secara default. Arsitektur penyimpanan Neptunus mengurangi biaya dan menghilangkan kebutuhan untuk penyediaan atau penyediaan penyimpanan yang berlebihan untuk menangani pertumbuhan data masa depan.

Sebelum memigrasikan data Anda ke Neptunus, ada baiknya Anda membiasakan diri dengan model data grafik properti Neptunus dan semantik transaksi.

Perbedaan operasional antara Neptunus dan Neo4j

Neptunus adalah layanan yang dikelola sepenuhnya yang mengotomatiskan banyak tugas operasional normal yang harus Anda lakukan saat menggunakan database on-premise atau self-managed seperti Neo4j Enterprise atau Community Edition:

- Pencadangan otomatis — Neptunus mencadangkan volume cluster Anda secara otomatis dan mempertahankan cadangan untuk periode retensi yang Anda tentukan (dari 1 hingga 35 hari). Pencadangan ini terus menerus dan bertahap, sehingga Anda dapat dengan cepat mengembalikan ke titik mana pun dalam periode retensi. Tidak ada dampak performa atau gangguan layanan basis data yang terjadi saat data cadangan ditulis.

- Snapshot Manual — Neptunus memungkinkan Anda membuat snapshot volume penyimpanan cluster DB Anda untuk mencadangkan seluruh cluster DB. Snapshot semacam ini kemudian dapat digunakan untuk memulihkan database, membuat salinannya, dan membagikannya di seluruh akun.
- Kloning — Neptunus mendukung fitur kloning yang memungkinkan Anda membuat klon database yang hemat biaya dengan cepat. Klon menggunakan copy-on-write protokol untuk hanya membutuhkan ruang tambahan minimal setelah dibuat. Kloning basis data adalah cara yang efektif untuk mencoba fitur atau peningkatan Neptunus baru tanpa gangguan pada cluster asal.
- Pemantauan — Neptunus menyediakan berbagai metode untuk memantau kinerja dan penggunaan cluster Anda, termasuk:
  - Status instans
  - Integrasi dengan Amazon CloudWatch dan AWS CloudTrail
  - Kemampuan log audit
  - Notifikasi peristiwa
  - Penandaan
- Keamanan — Neptunus menyediakan lingkungan yang aman secara default. Sebuah cluster berada dalam VPC pribadi yang menyediakan isolasi jaringan dari sumber daya lain. Semua lalu lintas dienkripsi melalui SSL, dan semua data dienkripsi saat istirahat menggunakan. AWS KMS

  Selain itu, Neptunus terintegrasi AWS Identity and Access Management dengan (IAM) untuk memberikan otentikasi. Dengan menentukan kunci kondisi IAM, Anda dapat menggunakan kebijakan IAM untuk memberikan kontrol akses halus atas tindakan data.

Perbedaan perkakas dan integrasi antara Neptunus dan Neo4j

Neptunus memiliki arsitektur yang berbeda untuk integrasi dan alat dari Neo4j, yang dapat memengaruhi arsitektur aplikasi Anda. Neptunus menggunakan sumber daya komputasi cluster untuk memproses kueri, tetapi memanfaatkan layanan best-in-class AWS lain untuk fungsionalitas seperti pencarian teks lengkap (menggunakan OpenSearch), ETL (menggunakan Glue), dan sebagainya. Untuk daftar lengkap integrasi ini, lihatIntegrasi Neptunus.

Penyediaan infrastruktur saat bermigrasi dari Neo4j ke Neptunus

Cluster Amazon Neptunus dibangun untuk skala dalam tiga dimensi: penyimpanan, kapasitas tulis, dan kapasitas baca. Bagian di bawah ini membahas opsi khusus untuk dipertimbangkan saat bermigrasi.

Penyimpanan penyediaan

Penyimpanan untuk setiap cluster Neptunus secara otomatis disediakan, tanpa biaya administrasi apa pun di pihak Anda. Ini mengubah ukuran secara dinamis dalam potongan 10 GB karena kebutuhan penyimpanan cluster meningkat. Akibatnya, tidak perlu memperkirakan dan menyediakan atau penyimpanan yang berlebihan untuk menangani pertumbuhan data di masa depan.

Penyediaan kapasitas tulis

Neptunus menyediakan contoh penulis tunggal yang dapat diskalakan secara vertikal ke ukuran instans apa pun yang tersedia di halaman harga Neptunus. Saat membaca dan menulis data ke instance penulis, semua transaksi sesuai dengan ACID, dengan isolasi data sebagaimana didefinisikan dalamTingkat Isolasi Transaksi di Neptune.

Memilih ukuran optimal untuk instance writer memerlukan menjalankan pengujian beban untuk menentukan ukuran instans optimal untuk beban kerja Anda. Setiap instance dalam Neptunus dapat diubah ukurannya kapan saja dengan memodifikasi kelas instans DB. Anda dapat memperkirakan ukuran instans awal berdasarkan konkurensi dan latensi kueri rata-rata seperti yang dijelaskan di bawah ini. Memperkirakan ukuran instans optimal saat menyediakan klaster Anda

Penyediaan kapasitas baca

Neptunus dibangun untuk menskalakan instance replika baca baik secara horizontal, dengan menambahkan hingga 15 di antaranya dalam sebuah cluster (atau lebih dalam database global Neptunus), dan secara vertikal ke ukuran instans apa pun yang tersedia di halaman harga Neptunus. Semua instance replika baca Neptunus menggunakan volume penyimpanan dasar yang sama, memungkinkan replikasi data transparan dengan jeda minimal.

Selain mengaktifkan penskalaan horizontal permintaan baca dalam cluster Neptunus, replika baca juga bertindak sebagai target failover untuk instance penulis untuk memungkinkan ketersediaan tinggi. Lihat Pedoman operasional dasar Amazon Neptunus saran tentang cara menentukan jumlah dan penempatan replika baca yang sesuai di klaster Anda.

Untuk aplikasi di mana konektivitas dan beban kerja tidak dapat diprediksi, Neptunus juga mendukung fitur auto-scaling yang dapat secara otomatis menyesuaikan jumlah replika Neptunus berdasarkan kriteria yang Anda tentukan.

Untuk menentukan ukuran dan jumlah instans replika baca yang optimal, diperlukan pengujian beban yang dijalankan untuk menentukan karakteristik beban kerja baca yang harus mereka dukung. Setiap instance dalam Neptunus dapat diubah ukurannya kapan saja dengan memodifikasi kelas instans DB. Anda dapat memperkirakan ukuran instans awal berdasarkan konkurensi dan latensi kueri rata-rata, seperti yang dijelaskan di bagian berikutnya.

Gunakan Neptunus Tanpa Server untuk menskalakan instance pembaca dan penulis secara otomatis sesuai kebutuhan

Meskipun seringkali bermanfaat untuk dapat memperkirakan kapasitas komputasi yang diperlukan oleh beban kerja yang Anda antisipasi, Anda dapat mengonfigurasi fitur Neptunus Tanpa Server untuk menskalakan kapasitas baca dan tulis ke atas dan ke bawah secara otomatis. Ini dapat membantu Anda memenuhi persyaratan puncak sambil juga menskalakan kembali secara otomatis saat permintaan menurun.

Memperkirakan ukuran instans optimal saat menyediakan klaster Anda

Estimasi ukuran instans optimal memerlukan mengetahui latensi kueri rata-rata di Neptunus, saat beban kerja Anda berjalan, serta jumlah kueri bersamaan yang sedang diproses. Perkiraan kasar ukuran instans dapat dihitung sebagai latensi kueri rata-rata dikalikan dengan jumlah kueri bersamaan. Ini memberi Anda jumlah rata-rata utas bersamaan yang diperlukan untuk menangani beban kerja.

Setiap vCPU dalam instance Neptunus dapat mendukung dua utas kueri bersamaan, jadi membagi utas dengan 2 memberikan jumlah v yang CPUs diperlukan, yang kemudian dapat dikorelasikan dengan ukuran instance yang sesuai pada halaman harga Neptunus. Contoh:

```
Average Query Latency:         30ms (0.03s)
Number of concurrent queries:  1000/second

Number of threads needed:      0.03 x 1000 = 30 threads
Number of vCPUs needed:        30 / 2 = 15 vCPUs
```

Menghubungkan ini dengan jumlah v CPUs dalam sebuah instance, kita melihat bahwa kita mendapatkan perkiraan kasar bahwa a `r5.4xlarge` akan menjadi contoh yang direkomendasikan

untuk dicoba untuk beban kerja ini. Perkiraan ini kasar dan hanya dimaksudkan untuk memberikan panduan awal tentang pemilihan ukuran instance. Aplikasi apa pun harus melalui latihan ukuran yang tepat untuk menentukan jumlah dan jenis instance yang sesuai untuk beban kerja.

Persyaratan memori juga harus diperhitungkan, serta persyaratan pemrosesan. Neptunus paling berkinerja ketika data yang diakses oleh kueri tersedia di cache kumpulan buffer memori utama. Menyediakan memori yang cukup juga dapat mengurangi I/O biaya secara signifikan.

Detail tambahan dan panduan tentang ukuran instance di cluster Neptunus dapat ditemukan di halaman. Mengukur instans DB dalam sebuah klaster Neptune DB

Migrasi data dari Neo4j ke Neptunus

Saat melakukan migrasi dari Neo4j ke Amazon Neptunus, memigrasikan data adalah langkah besar dalam prosesnya. Ada beberapa pendekatan untuk memigrasi data. Pendekatan yang benar ditentukan oleh kebutuhan aplikasi, ukuran data, dan jenis migrasi yang diinginkan. Namun, banyak dari migrasi ini semuanya memerlukan penilaian pertimbangan yang sama, yang beberapa disorot di bawah ini.

> **Note**
>
> Lihat Migrasi database grafik Neo4j ke Neptunus dengan utilitas yang sepenuhnya otomatis di Blog AWS Database untuk step-by-step panduan lengkap tentang salah satu contoh cara melakukan migrasi data offline.

Menilai migrasi data dari Neo4j ke Neptunus

Langkah pertama saat menilai migrasi data apa pun adalah menentukan bagaimana Anda akan memigrasikan data. Opsi bergantung pada arsitektur aplikasi yang dimigrasikan, ukuran data, dan kebutuhan ketersediaan selama migrasi. Secara umum, migrasi cenderung jatuh ke dalam salah satu dari dua kategori: online atau offline.

Migrasi offline cenderung menjadi yang paling sederhana untuk dilakukan, karena aplikasi tidak menerima lalu lintas baca atau tulis selama migrasi. Setelah aplikasi berhenti menerima lalu lintas, data dapat diekspor, dioptimalkan, diimpor, dan aplikasi diuji sebelum aplikasi diaktifkan kembali.

Migrasi online lebih kompleks, karena aplikasi masih perlu menerima lalu lintas baca dan tulis saat data sedang dimigrasikan. Kebutuhan yang tepat dari setiap migrasi online mungkin berbeda, tetapi arsitektur umum umumnya akan serupa dengan yang berikut:

- Umpan perubahan yang sedang berlangsung pada database perlu diaktifkan di Neo4j dengan mengonfigurasi Neo4j Streams sebagai sumber ke cluster Kafka.
- Setelah ini selesai, ekspor sistem yang sedang berjalan dapat diambil, mengikuti instruksiMengekspor data dari Neo4j saat bermigrasi ke Neptunus, dan waktu yang dicatat untuk korelasi selanjutnya dengan topik Kafka.
- Data yang diekspor kemudian diimpor ke Neptunus, mengikuti instruksi di. Mengimpor data dari Neo4j saat bermigrasi ke Neptunus

- Data yang diubah dari aliran Kafka kemudian dapat disalin ke cluster Neptunus menggunakan arsitektur yang mirip dengan yang dijelaskan dalam Menulis ke Amazon Neptunus dari Amazon Kinesis Data Streams. Perhatikan bahwa replikasi perubahan dapat dijalankan secara paralel untuk memvalidasi arsitektur dan kinerja aplikasi baru.
- Setelah migrasi data divalidasi, lalu lintas aplikasi dapat dialihkan ke cluster Neptunus dan instance Neo4j dapat dinonaktifkan.

Pengoptimalan model data untuk migrasi dari Neo4j ke Neptunus

Baik Neptunus dan Neo4j mendukung grafik properti berlabel (LPG). Namun, Neptunus memiliki beberapa perbedaan arsitektur dan model data yang dapat Anda manfaatkan untuk mengoptimalkan kinerja:

Mengoptimalkan node dan edge IDs

Neo4j secara otomatis menghasilkan panjang numerik. IDs Menggunakan Cypher Anda dapat merujuk ke node dengan ID, tetapi ini umumnya tidak disarankan untuk mencari node oleh properti yang diindeks.

Neptunus memungkinkan Anda untuk memasok berbasis IDs string Anda sendiri untuk simpul dan tepi. Jika Anda tidak memasok milik Anda sendiri IDs, Neptunus secara otomatis menghasilkan representasi UUIDs string untuk tepi dan simpul baru.

Jika Anda memigrasikan data dari Neo4j ke Neptunus dengan mengekspor dari Neo4j dan kemudian mengimpor massal ke Neptunus, Anda dapat mempertahankan Neo4j's. IDs Nilai numerik yang dihasilkan oleh Neo4j dapat bertindak sebagai disediakan pengguna saat IDs mengimpor ke Neptunus, di mana mereka direpresentasikan sebagai string daripada nilai numerik.

Namun, ada keadaan di mana Anda mungkin ingin mempromosikan properti simpul untuk menjadi ID simpul. Sama seperti mencari node menggunakan properti yang diindeks adalah cara tercepat untuk menemukan simpul di Neo4j, mencari simpul dengan ID adalah cara tercepat untuk menemukan simpul di Neptunus. Oleh karena itu, jika Anda dapat mengidentifikasi properti simpul yang sesuai yang berisi nilai unik, Anda harus mempertimbangkan untuk mengganti simpul ~id dengan nilai properti yang dinominasikan dalam file CSV pemuatan massal Anda. Jika Anda melakukan ini, Anda juga harus menulis ulang nilai ~from dan ~to edge yang sesuai dalam file CSV Anda.

Batasan skema saat memigrasikan data dari Neo4j ke Neptunus

Di dalam Neptunus, satu-satunya kendala skema yang tersedia adalah keunikan ID simpul atau tepi. Aplikasi yang perlu memanfaatkan kendala keunikan didorong untuk melihat pendekatan ini

untuk mencapai kendala keunikan melalui menentukan node atau ID tepi. Jika aplikasi menggunakan beberapa kolom sebagai kendala keunikan, ID dapat diatur ke kombinasi dari nilai-nilai ini. Misalnya, `id=123, code='SEA'` dapat direpresentasikan sebagai`ID='123_SEA'`) untuk mencapai kendala keunikan yang kompleks.

Optimalisasi arah tepi saat memigrasikan data dari Neo4j ke Neptunus

Ketika node, tepi, atau properti ditambahkan ke Neptunus, mereka secara otomatis diindeks dalam tiga cara berbeda, dengan indeks keempat opsional. Karena bagaimana Neptunus membangun dan menggunakan indeks, kueri yang mengikuti tepi keluar lebih efisien daripada yang menggunakan tepi masuk. Dalam hal model penyimpanan data grafik Neptunus, ini adalah pencarian berbasis subjek yang menggunakan indeks SPOG.

Jika, dalam memigrasikan model data dan kueri Anda ke Neptunus, Anda menemukan bahwa kueri terpenting Anda bergantung pada melintasi tepi yang masuk di mana ada tingkat penggemar yang tinggi, Anda mungkin ingin mempertimbangkan untuk mengubah model Anda sehingga lintasan ini mengikuti tepi keluar sebagai gantinya, terutama ketika Anda tidak dapat menentukan label tepi mana yang akan dilintasi. Untuk melakukannya, balikkan arah tepi yang relevan dan perbarui label tepi untuk mencerminkan semantik perubahan arah ini. Misalnya, Anda dapat mengubah:

```
person_A – parent_of – person_B
   to:
person_B – child_of – person_A
```

Untuk membuat perubahan ini dalam file CSV tepi beban massal, cukup tukar judul `~from` dan `~to` kolom, dan perbarui nilai kolom. `~label`

Sebagai alternatif untuk membalikkan arah tepi, Anda dapat mengaktifkan indeks Neptunus keempat, indeks OSGP, yang membuat melintasi tepi masuk, atau pencarian berbasis objek, jauh lebih efisien. Namun, mengaktifkan indeks keempat ini akan menurunkan tingkat penyisipan dan membutuhkan lebih banyak penyimpanan.

Optimasi pemfilteran saat memigrasikan data dari Neo4j ke Neptunus

Neptunus dioptimalkan untuk bekerja paling baik ketika properti disaring ke properti paling selektif yang tersedia. Ketika beberapa filter digunakan, kumpulan item yang cocok ditemukan untuk masing-masing dan kemudian tumpang tindih semua set yang cocok dihitung. Jika memungkinkan, menggabungkan beberapa properti ke dalam satu properti meminimalkan jumlah pencarian indeks dan mengurangi latensi kueri.

Misalnya, kueri ini menggunakan dua pencarian indeks dan gabungan:

```
MATCH (n) WHERE n.first_name='John' AND n.last_name='Doe' RETURN n
```

Kueri ini mengambil informasi yang sama menggunakan pencarian indeks tunggal:

```
MATCH (n) WHERE n.name='John Doe' RETURN n
```

Neptunus mendukung tipe data yang berbeda dari Neo4j.

Pemetaan tipe data Neo4j ke dalam tipe data yang didukung Neptunus

- Logis: `Boolean`

  Petakan ini di `Bool` Neptunus ke atau. `Boolean`
- Numerik: `Number`

  Petakan ini di Neptunus ke yang tersempit dari jenis OpenCypher Neptunus berikut yang dapat mendukung semua nilai properti numerik yang dimaksud:

  ```
  Byte
  Short
  Integer
  Long
  Float
  Double
  ```
- Teks: `String`

  Petakan ini di `String` Neptunus ke.
- Titik waktu:

  ```
  Date
  Time
  LocalTime
  DateTime
  LocalDateTime
  ```

  Petakan ini di `Date` Neptunus sebagai UTC, menggunakan salah satu format ISO-8601 berikut yang didukung Neptunus:

```
yyyy-MM-dd
yyyy-MM-ddTHH:mm
yyyy-MM-ddTHH:mm:ss
yyyy-MM-ddTHH:mm:ssZ
```

- Durasi waktu: `Duration`

  Petakan ini di Neptunus ke nilai numerik untuk aritmatika tanggal, jika perlu.

- Spasial: `Point`

  Petakan ini di Neptunus ke dalam nilai numerik komponen, yang masing-masing kemudian menjadi properti terpisah, atau ekspresikan sebagai nilai String untuk ditafsirkan oleh aplikasi klien. Perhatikan bahwa integrasi penelusuran teks lengkap Neptunus OpenSearch memungkinkan Anda mengindeks properti geolokasi.

Migrasi properti multivaluasi dari Neo4j ke Neptunus

Neo4j memungkinkan daftar homogen tipe sederhana untuk disimpan sebagai properti dari kedua node dan tepi. Daftar ini dapat berisi nilai duplikat.

Neptunus, bagaimanapun, hanya mengizinkan kardinalitas set atau tunggal untuk properti simpul, dan kardinalitas tunggal untuk properti tepi dalam data grafik properti. Akibatnya, tidak ada migrasi langsung properti daftar simpul Neo4j yang berisi nilai duplikat ke properti simpul Neptunus, atau properti daftar hubungan Neo4j ke properti tepi Neptunus.

Beberapa strategi yang mungkin untuk memigrasikan properti node multivaluasi Neo4j dengan nilai duplikat ke Neptunus adalah sebagai berikut:

- Buang nilai duplikat dan konversikan properti node Neo4j bernilai multivaluasi ke properti simpul Neptunus kardinalitas yang ditetapkan. Perhatikan bahwa himpunan Neptunus mungkin tidak mencerminkan urutan item dalam properti multivaluasi Neo4j asli.
- Ubah properti node Neo4j multivaluasi menjadi representasi string dari daftar berformat JSON di properti string simpul Neptunus.
- Ekstrak masing-masing nilai properti multivalued ke dalam simpul terpisah dengan properti nilai, dan hubungkan simpul tersebut ke simpul induk menggunakan tepi berlabel dengan nama properti.

Demikian pula, strategi yang mungkin untuk memigrasikan properti hubungan multivaluasi Neo4j ke Neptunus adalah sebagai berikut:

- Ubah properti hubungan Neo4j multivaluasi menjadi representasi string dari daftar berformat JSON dan simpan sebagai properti string tepi Neptunus.
- Memfaktorkan ulang hubungan Neo4j menjadi tepi Neptunus yang masuk dan keluar yang melekat pada simpul perantara. Ekstrak masing-masing nilai properti hubungan multivalued ke dalam simpul terpisah dengan properti nilai dan simpul tersebut ke simpul perantara ini menggunakan tepi berlabel dengan nama properti.

Perhatikan bahwa representasi string dari daftar berformat JSON tidak jelas terhadap bahasa kueri OpenCypher, meskipun OpenCypher menyertakan predikat yang memungkinkan pencarian sederhana di dalam nilai string. `CONTAINS`

Mengekspor data dari Neo4j saat bermigrasi ke Neptunus

Saat mengekspor data dari Neo4j, gunakan prosedur APOC untuk mengekspor ke CSV atau ke GraphMl. Meskipun dimungkinkan untuk mengekspor ke format lain, ada alat sumber terbuka untuk mengonversi data CSV yang diekspor dari Neo4j ke format beban massal Neptunus, dan juga alat sumber terbuka untuk mengonversi data GraphMl yang diekspor dari Neo4j ke format beban massal Neptunus.

Anda juga dapat mengekspor data langsung ke Amazon S3 menggunakan berbagai prosedur APOC. Mengekspor ke bucket Amazon S3 dinonaktifkan secara default, tetapi dapat diaktifkan menggunakan prosedur yang disorot dalam Mengekspor ke Amazon S3 dalam dokumentasi NEO4j APOC.

Mengimpor data dari Neo4j saat bermigrasi ke Neptunus

Anda dapat mengimpor data ke Neptunus baik dengan menggunakan pemuat massal Neptunus atau dengan menggunakan logika aplikasi dalam bahasa kueri yang didukung seperti OpenCypher.

Pemuat massal Neptunus adalah pendekatan yang lebih disukai untuk mengimpor data dalam jumlah besar karena memberikan kinerja impor yang dioptimalkan jika Anda mengikuti praktik terbaik. Pemuat massal mendukung dua format CSV yang berbeda, di mana data yang diekspor dari Neo4j dapat dikonversi menggunakan utilitas sumber terbuka yang disebutkan di atas di bagian ini. Mengekspor data

Anda juga dapat menggunakan OpenCypher untuk mengimpor data dengan logika khusus untuk mengurai, mengubah, dan mengimpor. Anda dapat mengirimkan kueri OpenCypher baik melalui titik akhir HTTPS (yang direkomendasikan) atau dengan menggunakan driver baut.

Migrasi aplikasi dari Neo4j ke Neptunus

Setelah Anda memigrasikan data Anda dari Neo4j ke Neptunus, langkah selanjutnya adalah memigrasikan aplikasi itu sendiri. Seperti halnya data, ada beberapa pendekatan untuk memigrasikan aplikasi Anda berdasarkan alat yang Anda gunakan, persyaratan, perbedaan arsitektur, dan sebagainya. Hal-hal yang biasanya perlu Anda pertimbangkan dalam proses ini diuraikan di bawah ini.

Migrasi koneksi saat berpindah dari Neo4j ke Neptunus

Jika saat ini Anda tidak menggunakan driver Bolt, atau ingin menggunakan alternatif, Anda dapat terhubung ke titik akhir HTTPS yang menyediakan akses penuh ke data yang dikembalikan.

Jika Anda memiliki aplikasi yang menggunakan protokol Bolt, Anda dapat memigrasikan koneksi ini ke Neptunus dan membiarkan aplikasi Anda terhubung menggunakan driver yang sama seperti yang Anda lakukan di Neo4j. Untuk terhubung ke Neptunus, Anda mungkin perlu membuat satu atau beberapa perubahan berikut pada aplikasi Anda:

- URL dan port perlu diperbarui untuk menggunakan titik akhir cluster dan port cluster (defaultnya adalah 8182).
- Neptunus membutuhkan semua koneksi untuk menggunakan SSL, jadi Anda perlu menentukan untuk setiap koneksi yang dienkripsi.
- Neptunus mengelola otentikasi melalui penetapan kebijakan dan peran IAM. Kebijakan dan peran IAM memberikan tingkat manajemen pengguna yang sangat fleksibel dalam aplikasi, jadi penting untuk membaca dan memahami informasi dalam ikhtisar IAM sebelum mengonfigurasi cluster Anda.
- Koneksi baut berperilaku berbeda di Neptunus daripada di Neo4j dalam beberapa cara, seperti yang dijelaskan dalam. Perilaku koneksi baut di Neptunus
- Anda dapat menemukan informasi dan saran lebih lanjut diPraktik Terbaik Neptunus Menggunakan OpenCypher dan Bolt.

Ada contoh kode untuk bahasa yang umum digunakan seperti Java, Python, .NET, dan NodeJS, dan untuk skenario koneksi seperti menggunakan otentikasi IAM, di. Menggunakan protokol Bolt untuk membuat kueri OpenCypher ke Neptunus

Merutekan kueri ke instance cluster saat berpindah dari Neo4j ke Neptunus

Aplikasi klien Neo4j menggunakan driver routing dan menentukan mode akses untuk merutekan permintaan baca dan tulis ke server yang sesuai dalam cluster kausal.

Saat memigrasikan aplikasi klien ke Neptunus, gunakan titik akhir Neptunus untuk merutekan kueri secara efisien ke instance yang sesuai di cluster Anda:

- Semua koneksi ke Neptunus harus `bolt://` menggunakan `bolt+routing://` bukan `neo4j://` atau di URL.
- Titik akhir cluster terhubung ke instance utama saat ini di cluster Anda. Gunakan titik akhir cluster untuk merutekan permintaan penulisan ke primer.
- Titik akhir pembaca mendistribusikan koneksi di seluruh instance read-replica di cluster Anda. Jika Anda memiliki kluster instans tunggal tanpa instance read-replica, titik akhir pembaca terhubung ke instance utama, yang mendukung operasi penulisan. Jika klaster memang berisi satu atau lebih instance replika baca, mengirim permintaan tulis ke titik akhir pembaca akan menghasilkan pengecualian.
- Setiap instance di cluster Anda juga dapat memiliki titik akhir instance-nya sendiri. Gunakan titik akhir instance jika aplikasi klien Anda perlu mengirim permintaan ke instance tertentu di cluster.

Untuk informasi selengkapnya, lihat Pertimbangan titik akhir Neptune.

Konsistensi data di Neptunus

Saat menggunakan cluster kausal Neo4j, replika baca pada akhirnya konsisten dengan server inti, tetapi aplikasi klien dapat memastikan konsistensi kausal dengan menggunakan rantai kausal. Chaining kausal memerlukan melewati bookmark antar transaksi, yang memungkinkan aplikasi klien untuk menulis ke server inti dan kemudian membaca tulisannya sendiri dari replika baca.

Di Neptunus, contoh replika baca akhirnya konsisten dengan penulis, dengan jeda replika yang biasanya kurang dari 100 milidetik. Namun, sampai perubahan telah direplikasi, pembaruan ke tepi dan simpul yang ada dan penambahan tepi dan simpul baru tidak terlihat pada contoh replika. Oleh karena itu, jika aplikasi Anda membutuhkan konsistensi langsung di Neptunus dengan membaca setiap penulisan, gunakan titik akhir cluster untuk operasi. read-after-write Ini adalah satu-satunya waktu untuk menggunakan titik akhir cluster untuk operasi baca. Dalam semua keadaan lain, gunakan titik akhir pembaca untuk membaca.

Memigrasi kueri dari Neo4j ke Neptunus

Meskipun dukungan Neptune untuk OpenCypher secara dramatis mengurangi jumlah pekerjaan yang diperlukan untuk memigrasikan kueri dari Neo4j, masih ada beberapa perbedaan untuk dinilai saat bermigrasi:

- Seperti dibahas di Pengoptimalan model data atas, mungkin ada modifikasi pada model data Anda yang perlu Anda buat untuk membuat model data grafik yang dioptimalkan untuk Neptunus, yang pada gilirannya akan memerlukan perubahan pada kueri dan pengujian Anda.
- Neo4j menawarkan berbagai ekstensi bahasa khusus Cypher yang tidak termasuk dalam spesifikasi OpenCypher yang diterapkan oleh Neptunus. Bergantung pada kasus penggunaan dan fitur yang digunakan, mungkin ada solusi dalam bahasa OpenCypher, atau menggunakan bahasa Gremlin, atau melalui mekanisme lain seperti yang dijelaskan dalam. Menulis ulang kueri Cypher untuk dijalankan di OpenCypher di Neptunus
- Aplikasi sering menggunakan komponen middleware lain untuk berinteraksi dengan database, bukan driver Bolt itu sendiri. Silakan periksa Kompatibilitas Neptunus dengan Neo4j untuk melihat apakah alat atau middleware yang Anda gunakan didukung.
- Dalam kasus failover, driver Bolt mungkin terus terhubung ke instance penulis atau pembaca sebelumnya karena titik akhir cluster yang disediakan untuk koneksi telah diselesaikan ke alamat IP. Penanganan kesalahan yang tepat dalam aplikasi Anda harus menangani ini, seperti yang dijelaskan dalamBuat koneksi baru setelah failover.
- Ketika transaksi dibatalkan karena konflik yang tidak dapat diselesaikan atau batas waktu tunggu kunci-tunggu, Neptunus merespons dengan file. `ConcurrentModificationException` Untuk informasi selengkapnya, lihat Kode Kesalahan Mesin. Sebagai praktik terbaik, klien harus selalu menangkap dan menangani pengecualian ini.

  A `ConcurrentModificationException` terjadi kadang-kadang ketika beberapa thread atau beberapa aplikasi menulis ke sistem secara bersamaan. Karena tingkat isolasi transaksi, konflik ini terkadang tidak dapat dihindari.
- Neptunus mendukung menjalankan kueri Gremlin dan OpenCypher pada data yang sama. Ini berarti bahwa dalam beberapa skenario Anda mungkin perlu mempertimbangkan untuk menggunakan Gremlin, dengan kemampuan kueri yang lebih kuat, untuk melakukan beberapa fungsionalitas kueri Anda.

Seperti dibahas di Penyediaan infrastruktur atas, setiap aplikasi harus melalui latihan ukuran yang tepat untuk memastikan bahwa jumlah instance, ukuran instance, dan topologi cluster semuanya dioptimalkan untuk beban kerja spesifik aplikasi.

Pertimbangan yang dibahas di sini untuk memigrasikan aplikasi Anda adalah yang paling umum, tetapi ini bukan daftar lengkap. Setiap aplikasi unik. Silakan hubungi untuk AWS mendukung atau melibatkan tim akun Anda jika Anda memiliki pertanyaan lebih lanjut.

Migrasi fitur dan alat yang khusus untuk Neo4j

Neo4j memiliki berbagai fitur khusus dan add-on dengan funsionalitas yang dapat diandalkan aplikasi Anda. Saat mengevaluasi kebutuhan untuk memigrasikan fungsi ini, sering kali membantu untuk menyelidiki apakah ada pendekatan yang lebih baik di dalam AWS untuk mencapai tujuan yang sama. Mempertimbangkan perbedaan arsitektur antara Neo4j dan Neptunus, Anda sering dapat menemukan alternatif efektif yang memanfaatkan layanan atau integrasi lain. AWS

Lihat Kompatibilitas Neptunus dengan Neo4j daftar fitur khusus Neo4J dan solusi yang disarankan.

Kompatibilitas Neptunus dengan Neo4j

Neo4j memiliki pendekatan all-in-one arsitektur, di mana pemuatan data, ETL data, kueri aplikasi, penyimpanan data, dan operasi manajemen semuanya terjadi dalam kumpulan sumber daya komputasi yang sama, seperti instance. EC2 Amazon Neptunus adalah database grafik spesifikasi terbuka yang berfokus pada OLTP di mana arsitektur memisahkan operasi dan memisahkan sumber daya sehingga dapat diskalakan secara dinamis.

Ada berbagai fitur dan perkakas di Neo4j, termasuk perkakas pihak ketiga, yang bukan bagian dari spesifikasi OpenCypher, tidak kompatibel dengan OpenCypher, atau tidak kompatibel dengan implementasi OpenCypher Neptune. Di bawah ini tercantum beberapa yang paling umum dari ini.

Fitur khusus Neo4J tidak ada di Neptunus

- **`LOAD CSV`**Neptunus memiliki pendekatan arsitektur yang berbeda untuk memuat data dari Neo4j. Untuk memungkinkan penskalaan dan pengoptimalan biaya yang lebih baik, Neptunus menerapkan pemisahan kekhawatiran seputar sumber daya, dan merekomendasikan penggunaan salah AWS satu integrasi layananAWS Glue seperti untuk melakukan proses ETL yang diperlukan untuk menyiapkan data dalam format yang didukung oleh pemuat massal Neptunus.

  Pilihan lain adalah melakukan hal yang sama menggunakan kode aplikasi yang berjalan pada sumber daya AWS komputasi seperti EC2 instans Amazon, fungsi Lambda, Amazon Elastic Container Service, AWS Batch pekerjaan, dan sebagainya. Kode dapat menggunakan titik akhir HTTPS Neptunus atau titik akhir Bolt.
- Kontrol akses berbutir halus - Neptunus mendukung kontrol akses granular atas tindakan akses data menggunakan kunci kondisi IAM. Kontrol akses berbutir halus tambahan dapat diimplementasikan pada lapisan aplikasi.
- Neo4j Fabric - Neptunus mendukung federasi kueri di seluruh database untuk beban kerja RDF menggunakan kata kunci SPARQL. `SERVICE` Karena saat ini tidak ada standar atau spesifikasi terbuka untuk federasi kueri untuk beban kerja grafik properti, fungsionalitas itu perlu diimplementasikan pada lapisan aplikasi.
- Kontrol akses berbasis peran (RBAC) - Neptunus mengelola otentikasi melalui penetapan kebijakan dan peran IAM. Kebijakan dan peran IAM memberikan tingkat manajemen pengguna yang sangat fleksibel dalam suatu aplikasi, jadi ada baiknya membaca dan memahami informasi dalam ikhtisar IAM sebelum mengonfigurasi cluster Anda.
- Bookmark — Cluster Neptunus terdiri dari satu contoh penulis dan hingga 15 contoh replika baca. Data yang ditulis ke contoh penulis sesuai dengan ACID dan memberikan jaminan konsistensi

yang kuat pada pembacaan berikutnya. Replicas baca menggunakan volume penyimpanan yang sama dengan instance penulis dan pada akhirnya konsisten, biasanya dalam waktu kurang dari 100 ms dari waktu data ditulis. Jika kasus penggunaan Anda memiliki kebutuhan mendesak untuk menjamin konsistensi baca penulisan baru, pembacaan ini harus diarahkan ke titik akhir cluster alih-alih titik akhir pembaca.

- Prosedur APOC — Karena prosedur APOC tidak termasuk dalam spesifikasi OpenCypher, Neptunus tidak memberikan dukungan langsung untuk prosedur eksternal. Sebaliknya, Neptunus mengandalkan integrasi dengan layanan AWS lain untuk mencapai fungsionalitas pengguna akhir yang serupa dengan cara yang terukur, aman, dan kuat. Terkadang prosedur APOC dapat ditulis ulang dalam OpenCypher atau Gremlin, dan beberapa tidak relevan dengan aplikasi Neptunus.

  Secara umum, prosedur APOC termasuk dalam kategori di bawah ini:

  - Impor — Neptunus mendukung pengimporan data menggunakan berbagai format menggunakan bahasa kueri, pemuat massal Neptunus, atau sebagai target. AWS Database Migration Service Operasi ETL pada data dapat dilakukan dengan menggunakan AWS Glue dan paket `neptune-python-utils`open-source.
  - Ekspor - Neptunus mendukung ekspor data menggunakan `neptune-export`utilitas, yang mendukung berbagai format dan metode ekspor umum.
  - Integrasi Database — Neptunus mendukung integrasi dengan database lain menggunakan alat ETL seperti atau alat migrasi AWS Glue seperti. AWS Database Migration Service
  - Pembaruan Grafik - Neptunus mendukung serangkaian fitur yang kaya untuk memperbarui data grafik properti melalui dukungannya untuk bahasa kueri OpenCypher dan Gremlin. Lihat Cypher menulis ulang contoh penulisan ulang prosedur yang umum digunakan.
  - Struktur Data — Neptunus mendukung serangkaian fitur yang kaya untuk memperbarui data grafik properti melalui dukungannya untuk bahasa kueri OpenCypher dan Gremlin. Lihat Cypher menulis ulang contoh penulisan ulang prosedur yang umum digunakan.
  - Temporal (Date Time) - Neptunus mendukung serangkaian fitur yang kaya untuk memperbarui data grafik properti melalui dukungannya untuk bahasa kueri OpenCypher dan Gremlin. Lihat Cypher menulis ulang contoh penulisan ulang prosedur yang umum digunakan.
  - Matematika — Neptunus mendukung serangkaian fitur yang kaya untuk memperbarui data grafik properti melalui dukungannya untuk bahasa kueri OpenCypher dan Gremlin. Lihat Cypher menulis ulang contoh penulisan ulang prosedur yang umum digunakan.
  - Advanced Graph Querying — Neptunus mendukung serangkaian fitur yang kaya untuk memperbarui data grafik properti melalui dukungannya untuk bahasa kueri OpenCypher dan Gremlin. Lihat Cypher menulis ulang contoh penulisan ulang prosedur yang umum digunakan.

  - Membandingkan Grafik — Neptunus mendukung serangkaian fitur yang kaya untuk memperbarui data grafik properti melalui dukungannya untuk bahasa kueri OpenCypher dan Gremlin. Lihat Cypher menulis ulang contoh penulisan ulang prosedur yang umum digunakan.
  - Eksekusi Cypher — Neptunus mendukung serangkaian fitur yang kaya untuk memperbarui data grafik properti melalui dukungannya untuk bahasa kueri OpenCypher dan Gremlin. Lihat Cypher menulis ulang contoh penulisan ulang prosedur yang umum digunakan.
- Prosedur khusus - Neptunus tidak mendukung prosedur khusus yang dibuat oleh pengguna. Fungsionalitas ini harus diimplementasikan pada lapisan aplikasi.
- Geospasial — Meskipun Neptunus tidak memberikan dukungan asli untuk fitur geospasial, fungsionalitas serupa dapat dicapai melalui integrasi dengan layanan AWS lain, seperti yang ditunjukkan dalam posting blog ini: Gabungkan Amazon Neptunus dan OpenSearch Amazon Service untuk kueri geospasial oleh Ross Gabay dan Abhilash Vinod (1 Februari 2022).
- Ilmu Data Grafik — Neptunus mendukung analisis grafik hari ini melalui Neptunus Analytics, mesin yang dioptimalkan untuk memori yang mendukung perpustakaan algoritme analitik grafik.

  Neptunus juga menyediakan integrasi dengan AWS SDK Pandas dan beberapa contoh notebook yang menunjukkan bagaimana memanfaatkan integrasi ini dalam lingkungan Python untuk menjalankan analitik pada data grafik.

- Batasan Skema — Dalam Neptunus, satu-satunya kendala skema yang tersedia adalah keunikan ID node atau tepi. Tidak ada fitur untuk menentukan batasan skema tambahan, atau keunikan tambahan atau batasan nilai pada elemen dalam grafik. Nilai ID di Neptunus adalah string dan dapat diatur menggunakan Gremlin, seperti ini:

  ```
  g.addV('person').property(id, '1') )
  ```

  Aplikasi yang perlu memanfaatkan ID sebagai kendala keunikan didorong untuk mencoba pendekatan ini untuk mencapai kendala keunikan. Jika aplikasi menggunakan beberapa kolom sebagai kendala keunikan, ID dapat diatur ke kombinasi dari nilai-nilai ini. Misalnya `id=123, code='SEA'` dapat direpresentasikan `ID='123_SEA'` untuk mencapai kendala keunikan yang kompleks.
- Multi-tenancy — Neptunus hanya mendukung satu grafik per cluster. Untuk membangun sistem multi-tenant menggunakan Neptunus, gunakan beberapa cluster atau partisi penyewa secara logis dalam satu grafik dan gunakan logika sisi aplikasi untuk menegakkan pemisahan. Misalnya, tambahkan properti `tenantId` dan sertakan di setiap kueri, seperti ini:

```
MATCH p=(n {tenantId:1})-[]->({tenantId:1}) RETURN p LIMIT 5)
```

Neptunus Tanpa Server membuatnya relatif mudah untuk menerapkan multi-tenancy menggunakan beberapa cluster DB, yang masing-masing diskalakan secara independen dan otomatis sesuai kebutuhan.

Dukungan Neptunus untuk alat Neo4j

Neptunus menyediakan alternatif berikut untuk alat Neo4j:

- Neo4j Browser — Neptunus menyediakan notebook grafik open-source yang menyediakan IDE yang berfokus pada pengembang untuk menjalankan kueri dan memvisualisasikan hasilnya.
- Neo4j Bloom — Neptunus mendukung visualisasi grafik kaya menggunakan solusi visualisasi pihak ketiga seperti Graph-explorer, Tom Sawyer, Cambridge Intelligence, Graphistry, metaphacts, dan G.V ().
- GraphQL — Neptunus saat ini mendukung GraphQL melalui integrasi kustom. AWS AppSync Lihat aplikasi Membangun grafik dengan posting blog Amazon Neptunus AWS dan Amplify, dan contoh proyek, Membangun aplikasi pelacak Kalori Tanpa Server dengan dan Amazon Neptunus. AWS AppSync
- NeoSemantics— Neptunus secara native mendukung model data RDF, sehingga pelanggan yang ingin menjalankan beban kerja RDF disarankan untuk menggunakan dukungan model RDF Neptunus.
- Arrows.app — Cypher yang dibuat saat mengekspor model menggunakan perintah ekspor kompatibel dengan Neptunus.
- Linkurious Ogma - Integrasi sampel dengan Linkurious Ogma tersedia di sini.
- Data Musim Semi Neo4j — Ini saat ini tidak kompatibel dengan Neptunus.
- Neo4j Spark Connector - Konektor percikan Neo4j dapat digunakan dalam Spark Job untuk terhubung ke Neptunus menggunakan OpenCypher. Berikut adalah beberapa contoh kode dan konfigurasi aplikasi:

  Contoh kode:

```
SparkSession spark = SparkSession
            .builder()
            .config("encryption.enabled", "true")
```

```
            .appName("Simple Application").config("spark.master",
 "local").getOrCreate();

Dataset<Row> df = spark.read().format("org.neo4j.spark.DataSource")
            .option("url", "bolt://(your cluster endpoint):8182")
            .option("encryption.enabled", "true")
            .option("query", "MATCH (n:airport) RETURN n")
            .load();

System.out.println("TOTAL RECORD COUNT: " + df.count());
spark.stop();
```

Konfigurasi aplikasi:

```
<dependency>
    <groupId>org.neo4j</groupId>
    <artifactId>neo4j-connector-apache-spark_2.12-4.1.0</artifactId>
    <version>4.0.1_for_spark_3</version>
</dependency>
```

Fitur dan alat Neo4j tidak tercantum di sini

Jika Anda menggunakan alat atau fitur yang tidak tercantum di sini, kami tidak yakin kompatibilitasnya dengan Neptunus atau layanan lain di dalamnya. AWS Silakan hubungi untuk AWS mendukung atau melibatkan tim akun Anda jika Anda memiliki pertanyaan lebih lanjut.

Menulis ulang kueri Cypher untuk dijalankan di OpenCypher di Neptunus

Bahasa OpenCypher adalah bahasa query deklaratif untuk grafik properti yang awalnya dikembangkan oleh Neo4j, kemudian open-source pada tahun 2015, dan berkontribusi pada proyek OpenCypher di bawah lisensi open-source Apache 2. Di AWS, kami percaya bahwa open source baik untuk semua orang dan kami berkomitmen untuk membawa nilai open source kepada pelanggan kami, dan keunggulan operasional AWS untuk komunitas open source.

OpenCypher sintaks didokumentasikan dalam Referensi Bahasa Kueri Cypher, Versi 9.

Karena OpenCypher berisi subset sintaks dan fitur bahasa kueri Cypher, beberapa skenario migrasi memerlukan penulisan ulang kueri dalam formulir yang sesuai dengan OpenCypher atau memeriksa metode alternatif untuk mencapai fungsionalitas yang diinginkan.

Bagian ini berisi rekomendasi untuk menangani perbedaan umum, tetapi sama sekali tidak lengkap. Anda harus menguji aplikasi apa pun menggunakan penulisan ulang ini secara menyeluruh untuk memastikan bahwa hasilnya sesuai dengan yang Anda harapkan.

Menulis ulang**None**,**All**, dan fungsi **Any** predikat

Fungsi-fungsi ini bukan bagian dari spesifikasi OpenCypher. Hasil yang sebanding dapat dicapai di OpenCypher menggunakan List Comprehension.

Misalnya, temukan semua jalur yang berpindah dari node `Start` ke node`End`, tetapi tidak ada perjalanan yang diizinkan melewati node dengan properti kelas`D`:

```
# Neo4J Cypher code
match p=(a:Start)-[:HOP*1..]->(z:End)
where none(node IN nodes(p) where node.class ='D')
return p

# Neptune openCypher code
match p=(a:Start)-[:HOP*1..]->(z:End)
where size([node IN nodes(p) where node.class = 'D']) = 0
return p
```

Pemahaman daftar dapat mencapai hasil ini sebagai berikut:

```
all  => size(list_comprehension(list)) = size(list)
any  => size(list_comprehension(list)) >= 1
none => size(list_comprehension(list)) = 0
```

Menulis ulang fungsi Cypher di OpenCypher **`reduce()`**

`reduce()`Fungsi ini bukan bagian dari spesifikasi OpenCypher. Hal ini sering digunakan untuk membuat agregasi data dari elemen dalam daftar. Dalam banyak kasus, Anda dapat menggunakan kombinasi Pemahaman Daftar dan `UNWIND` klausa untuk mencapai hasil yang serupa.

Misalnya, kueri Cypher berikut menemukan semua bandara di jalur yang memiliki satu hingga tiga pemberhentian antara Anchorage (ANC) dan Austin (AUS), dan mengembalikan jarak total setiap jalur:

```
MATCH p=(a:airport {code: 'ANC'})-[r:route*1..3]->(z:airport {code: 'AUS'})
RETURN p, reduce(totalDist=0, r in relationships(p) | totalDist + r.dist) AS totalDist
ORDER BY totalDist LIMIT 5
```

Anda dapat menulis kueri yang sama di OpenCypher untuk Neptunus sebagai berikut:

```
MATCH p=(a:airport {code: 'ANC'})-[r:route*1..3]->(z:airport {code: 'AUS'})
UNWIND [i in relationships(p) | i.dist] AS di
RETURN p, sum(di) AS totalDist
ORDER BY totalDist
LIMIT 5
```

Menulis ulang klausa Cypher FOREACH di OpenCypher

Klausa FOREACH bukan bagian dari spesifikasi OpenCypher. Hal ini sering digunakan untuk memperbarui data di tengah-tengah query, sering dari agregasi atau elemen dalam jalur.

Sebagai contoh jalur, temukan semua bandara di jalur dengan tidak lebih dari dua pemberhentian antara Anchorage (ANC) dan Austin (AUS) dan tetapkan properti yang dikunjungi di masing-masing bandara:

```
# Neo4J Example
MATCH p=(:airport {code: 'ANC'})-[*1..2]->({code: 'AUS'})
FOREACH (n IN nodes(p) | SET n.visited = true)

# Neptune openCypher
MATCH p=(:airport {code: 'ANC'})-[*1..2]->({code: 'AUS'})
WITH nodes(p) as airports
UNWIND airports as a
SET a.visited=true
```

Contoh lain adalah:

```
# Neo4J Example
MATCH p=(start)-[*]->(finish)
WHERE start.name = 'A' AND finish.name = 'D'
FOREACH (n IN nodes(p) | SET n.marked = true)

# Neptune openCypher
MATCH p=(start)-[*]->(finish)
WHERE start.name = 'A' AND finish.name = 'D'
UNWIND nodes(p) AS n
SET n.marked = true
```

Menulis ulang prosedur APOC Neo4j di Neptunus

Contoh di bawah ini menggunakan OpenCypher untuk menggantikan beberapa prosedur APOC yang paling umum digunakan. Contoh-contoh ini hanya untuk referensi, dan dimaksudkan untuk memberikan beberapa saran tentang cara menangani skenario umum. Dalam praktiknya, setiap aplikasi berbeda, dan Anda harus membuat strategi Anda sendiri untuk menyediakan semua fungsionalitas yang Anda butuhkan.

Prosedur penulisan ulang **`apoc.export`**

Neptunus menyediakan berbagai opsi untuk grafik penuh dan ekspor berbasis kueri dalam berbagai format output seperti CSV dan JSON, menggunakan utilitas ekspor neptunus (lihat). Mengekspor data dari cluster DB Neptunus

Prosedur penulisan ulang **`apoc.schema`**

Neptunus tidak memiliki skema, indeks, atau kendala yang didefinisikan secara eksplisit, sehingga banyak prosedur tidak lagi diperlukan. `apoc.schema` Contohnya adalah:

- `apoc.schema.assert`
- `apoc.schema.node.constraintExists`
- `apoc.schema.node.indexExists`,
- `apoc.schema.relationship.constraintExists`
- `apoc.schema.relationship.indexExists`
- `apoc.schema.nodes`
- `apoc.schema.relationships`

Neptunus OpenCypher memang mendukung pengambilan nilai yang serupa dengan yang dilakukan prosedur, seperti yang ditunjukkan di bawah ini, tetapi dapat mengalami masalah kinerja pada grafik yang lebih besar karena hal itu memerlukan pemindaian sebagian besar grafik untuk mengembalikan jawabannya.

```
# openCypher replacement for apoc.schema.properties.distinct
MATCH (n:airport)
RETURN DISTINCT n.runways
```

```
# openCypher replacement for apoc.schema.properties.distinctCount
MATCH (n:airport)
RETURN DISTINCT n.runways, count(n.runways)
```

Alternatif untuk **`apoc.do`** prosedur

Prosedur ini digunakan untuk menyediakan eksekusi kueri bersyarat yang mudah diimplementasikan menggunakan klausa OpenCypher lainnya. Di Neptunus setidaknya ada dua cara untuk mencapai perilaku serupa:

- Salah satu caranya adalah dengan menggabungkan kemampuan Pemahaman Daftar OpenCypher dengan klausa. `UNWIND`
- Cara lain adalah dengan menggunakan langkah choose () dan coalesce () di Gremlin.

Contoh pendekatan ini ditunjukkan di bawah ini.

Alternatif untuk apoc.do.when

```
# Neo4J Example
MATCH (n:airport {region: 'US-AK'})
CALL apoc.do.when(
 n.runways>=3,
 'SET n.is_large_airport=true RETURN n',
 'SET n.is_large_airport=false RETURN n',
 {n:n}
) YIELD value
WITH collect(value.n) as airports
RETURN size([a in airports where a.is_large_airport]) as large_airport_count,
size([a in airports where NOT a.is_large_airport]) as small_airport_count
```

```
# Neptune openCypher
MATCH (n:airport {region: 'US-AK'})
WITH n.region as region, collect(n) as airports
WITH [a IN airports where a.runways >= 3] as large_airports,
[a IN airports where a.runways < 3] as small_airports, airports
UNWIND large_airports as la
SET la.is_large_airport=true
WITH DISTINCT small_airports, airports
UNWIND small_airports as la
    SET la.small_airports=true
WITH DISTINCT airports
RETURN size([a in airports where a.is_large_airport]) as large_airport_count,
size([a in airports where NOT a.is_large_airport]) as small_airport_count

#Neptune Gremlin using choose()
g.V().
  has('airport', 'region', 'US-AK').
  choose(
    values('runways').is(lt(3)),
    property(single, 'is_large_airport', false),
    property(single, 'is_large_airport', true)).
  fold().
  project('large_airport_count', 'small_airport_count').
    by(unfold().has('is_large_airport', true).count()).
    by(unfold().has('is_large_airport', false).count())

 #Neptune Gremlin using coalesce()
g.V().
  has('airport', 'region', 'US-AK').
  coalesce(
    where(values('runways').is(lt(3))).
    property(single, 'is_large_airport', false),
    property(single, 'is_large_airport', true)).
  fold().
  project('large_airport_count', 'small_airport_count').
    by(unfold().has('is_large_airport', true).count()).
    by(unfold().has('is_large_airport', false).count())
```

Alternatif untuk apoc.do.case

```
# Neo4J Example
MATCH (n:airport {region: 'US-AK'})
CALL apoc.case([
```

```
 n.runways=1, 'RETURN "Has one runway" as b',
 n.runways=2, 'RETURN "Has two runways" as b'
 ],
 'RETURN "Has more than 2 runways" as b'
) YIELD value
RETURN {type: value.b,airport: n}

# Neptune openCypher
MATCH (n:airport {region: 'US-AK'})
WITH n.region as region, collect(n) as airports
WITH [a IN airports where a.runways =1] as single_runway,
[a IN airports where a.runways =2] as double_runway,
[a IN airports where a.runways >2] as many_runway
UNWIND single_runway as sr
    WITH {type: "Has one runway",airport: sr} as res, double_runway, many_runway
WITH DISTINCT double_runway as double_runway, collect(res) as res, many_runway
UNWIND double_runway as dr
    WITH {type: "Has two runways",airport: dr} as two_runways, res, many_runway
WITH collect(two_runways)+res as res, many_runway
UNWIND many_runway as mr
    WITH {type: "Has more than 2 runways",airport: mr} as res2, res, many_runway
WITH collect(res2)+res as res
UNWIND res as r
RETURN r

#Neptune Gremlin using choose()
g.V().
  has('airport', 'region', 'US-AK').
  project('type', 'airport').
    by(
      choose(values('runways')).
        option(1, constant("Has one runway")).
        option(2, constant("Has two runways")).
        option(none, constant("Has more than 2 runways"))).
    by(elementMap())

 #Neptune Gremlin using coalesce()
 g.V().
  has('airport', 'region', 'US-AK').
  project('type', 'airport').
    by(
      coalesce(
        has('runways', 1).constant("Has one runway"),
        has('runways', 2).constant("Has two runways"),
```

```
        constant("Has more than 2 runways"))).
    by(elementMap())
```

Alternatif untuk properti berbasis daftar

Neptunus saat ini tidak mendukung penyimpanan properti berbasis Daftar. Namun, hasil serupa dapat diperoleh dengan menyimpan nilai daftar sebagai string yang dipisahkan koma dan kemudian menggunakan `split()` fungsi `join()` and untuk membangun dan mendekonstruksi properti list.

Misalnya, jika kita ingin menyimpan daftar tag sebagai properti, kita dapat menggunakan contoh penulisan ulang yang menunjukkan cara mengambil properti yang dipisahkan koma dan kemudian menggunakan `join()` fungsi `split()` dan dengan Pemahaman Daftar untuk mencapai hasil yang sebanding:

```
# Neo4j Example (In this example, tags is a durable list of string.
MATCH (person:person {name: "TeeMan"})
WITH person, [tag in person.tags WHERE NOT (tag IN ['test1', 'test2', 'test3'])] AS
 newTags
SET person.tags = newTags
RETURN person

# Neptune openCypher
MATCH (person:person {name: "TeeMan"})
WITH person, [tag in split(person.tags, ',') WHERE NOT (tag IN ['test1', 'test2',
 'test3'])] AS newTags
SET person.tags = join(newTags,',')
RETURN person
```

Menulis ulang subkueri CALL

Subquery `CALL` Neptunus tidak mendukung `CALL (friend) { ... }` sintaks untuk mengimpor variabel ke dalam lingkup subquery (, dalam contoh ini). `friend` Silakan gunakan `WITH` klausa di dalam subquery untuk hal yang sama, misalnya,. `CALL { WITH friend ... }`

`CALL`Subquery opsional tidak didukung saat ini.

Perbedaan lain antara Neptunus OpenCypher dan Cypher

- Neptunus hanya mendukung koneksi TCP untuk protokol Bolt. WebSocketskoneksi untuk Bolt tidak didukung.

- Neptune OpenCypher menghapus spasi seperti yang didefinisikan oleh Unicode di, dan fungsi. `trim()` `ltrim()` `rtrim()`
- Di Neptunus OpenCypher`tostring(, )` ganda tidak secara otomatis beralih ke notasi E untuk nilai ganda yang besar.

Sumber daya untuk bermigrasi dari Neo4j ke Neptunus

Neptunus menyediakan beberapa alat dan sumber daya yang dapat membantu dalam proses migrasi.

Alat untuk membantu bermigrasi dari Neo4j ke Neptunus

- OpenCypher CheatSheet.
- neo4 j-to-neptune — Utilitas baris perintah untuk memigrasikan data dari Neo4j ke Neptunus. Alat ini mencakup kemampuan untuk:
  - Ekspor data dari grafik Neo4j yang dikonfigurasi dengan benar.
  - Ubah data itu menjadi format Neptunus.
  - Massal memuat data itu ke Neptunus.
  - Lakukan beberapa konversi dasar data selama konversi ke format Neptunus, seperti mengganti nama label simpul atau tepi dan menghasilkan elemen.
  - Hasilkan properti untuk node dan tepi menggunakan templat (misalnya, buat `~id` nilai menggunakan templat seperti `Person_{personid}` untuk situasi di mana Anda perlu membuat pengidentifikasi unik untuk elemen).
- OpenCypher Query Compatibility Checker — Alat ini mengambil masukan dari query OpenCypher dan akan:
  - Periksa kompatibilitas dengan versi Neptunus yang dipilih.
  - Identifikasi fungsi dan klausa tertentu yang tidak didukung dengan posisinya.
  - Sarankan penggantian jika tersedia.
  - Berikan deskripsi kesalahan dari kesalahan sintaks lainnya.

Migrasi grafik yang ada dari server Apache TinkerPop Gremlin ke Amazon Neptunus

Jika Anda memiliki data grafik di Server Apache TinkerPop Gremlin yang ingin Anda migrasi ke Amazon Neptunus, Anda akan mengambil langkah-langkah berikut:

1. Ekspor data dari server Gremlin ke Amazon Simple Storage Service (Amazon S3).
2. Mengkonversi data yang diekspor ke Format CSV yang dapat diimpor oleh pemuat massal Neptune.
3. Menggunakan Bulk Loader Neptune, impor data ke dalam klaster DB Neptune yang telah Anda siapkan.
4. Ubah aplikasi Anda yang sudah ada untuk terhubung ke titik akhir Gremlin Neptunus, dan buat perubahan yang diperlukan agar sesuai dengan Perbedaan implementasi Neptune Gremlin.

Migrasi grafik yang ada dari toko triple RDF ke Amazon Neptune

Jika Anda memiliki data grafik RDF/SPARQL untuk bermigrasi ke Amazon Neptunus, Anda akan mengambil langkah-langkah berikut:

1. Ekspor data dari toko triple RDF Anda.
2. Mengkonversi data yang diekspor ke Format yang dapat diimpor oleh pemuat massal Neptune.
3. Menyimpan data yang akan diimpor di Amazon Simple Storage Service (Amazon S3).
4. Menggunakan Bulk Loader Neptune, impor data dari Amazon S3 ke klaster DB Neptune yang telah Anda siapkan.
5. Ubah aplikasi Anda yang sudah ada untuk terhubung ke titik akhir SPARQL Neptunus.

Jika Anda ingin mencoba memigrasi data CSV grafik properti ke RDF, Anda dapat menggunakan Amazon Neptune CSV ke konverter RDF.

Menggunakan AWS Database Migration Service (AWS DMS) untuk bermigrasi dari database relasional atau NoSQL ke Amazon Neptunus

AWS Database Migration Service (AWS DMS) adalah layanan cloud yang memudahkan migrasi database relasional, gudang data, database NoSQL, dan jenis penyimpanan data lainnya. Jika Anda memiliki data grafik yang disimpan di salah satu database relasional atau NoSQL AWS DMS yang mendukung AWS DMS , dapat membantu Anda bermigrasi ke Neptunus dengan cepat dan aman, tanpa memerlukan waktu henti dari database Anda saat ini. Lihat Menggunakan AWS Database Migration Service untuk memuat data ke Amazon Neptunus dari penyimpanan data yang berbeda untuk detail.

Dataflow migrasi menggunakan AWS DMS adalah sebagai berikut:

- Buat objek AWS DMS pemetaan tabel. Objek JSON ini menentukan tabel yang harus dibaca dari database sumber Anda dan dalam urutan apa, dan bagaimana kolom mereka diberi nama. Hal ini juga dapat menyaring baris yang disalin dan memberikan transformasi nilai sederhana seperti mengonversi ke huruf kecil atau pembulatan.
- Buat Neptune `GraphMappingConfig` untuk menentukan bagaimana data yang diekstrak dari database sumber harus dimuat ke Neptune.
  - Untuk data RDF (dikueri menggunakan SPARQL), `GraphMappingConfig` ditulis dalam bahasa pemetaan R2RML standar W3.
  - Untuk data grafik properti (dikueri menggunakan Gremlin), `GraphMappingConfig` adalah objek JSON, seperti dijelaskan dalam GraphMappingConfig Tata Letak untuk Property-Graph/Data Gremlin.
- Buat instance AWS DMS replikasi di VPC yang sama dengan cluster DB Neptunus Anda, untuk melakukan migrasi.
- Buat bucket Amazon S3 untuk digunakan sebagai penyimpanan menengah untuk penahapan data yang sedang dimigrasi.
- Jalankan tugas AWS DMS migrasi.

Lihat Menggunakan AWS Database Migration Service untuk memuat data ke Amazon Neptunus dari penyimpanan data yang berbeda untuk detailnya, dan juga posting blog Chris Smith's four-piece, “Mengisi grafik Anda di Amazon Neptune dari database relasional menggunakan Database Migration Service (DMS) AWS :”

- Bagian 1: Mengatur panggung
- Bagian 2: Merancang model grafik properti
- Bagian 3: Merancang Model RDF
- Bagian 4: Menyatukan semuanya

Migrasi dari Blazegraph ke Amazon Neptune

Jika Anda memiliki grafik di sumber terbuka Blazegraf RDF triplestore, Anda dapat bermigrasi ke data grafik Anda ke Amazon Neptune menggunakan langkah-langkah berikut:

- AWS Infrastruktur penyediaan. Mulailah dengan menyediakan infrastruktur Neptunus yang diperlukan menggunakan templat (lihat). AWS CloudFormation Buat cluster Neptunus
- Ekspor data dari Blazegraph. Ada dua metode utama untuk mengekspor data dari Blazegraph, yaitu menggunakan kueri SPARQL CONSTRUCT atau menggunakan utilitas Ekspor Blazegraph.
- Impor data ke Neptunus. Anda kemudian dapat memuat file data yang diekspor ke Neptune menggunakan Neptune Workbench dan Bulk Loader Neptune.

Pendekatan ini juga umumnya berlaku untuk migrasi dari database triplestore RDF lainnya.

Kompatibilitas Blazegraph ke Neptune

Sebelum memigrasikan data grafik Anda ke Neptune, ada beberapa perbedaan signifikan antara Blazegraph dan Neptune yang harus Anda sadari. Perbedaan ini dapat memerlukan perubahan pada kueri, arsitektur aplikasi, atau keduanya, atau bahkan membuat migrasi tidak praktis:

- **`Full-text search`** – Dalam Blazegraph, Anda dapat menggunakan pencarian teks lengkap internal atau kemampuan pencarian teks lengkap eksternal melalui integrasi dengan Apache Solr. Jika Anda menggunakan salah satu fitur ini, tetaplah mendapatkan informasi tentang pembaruan terbaru pada fitur pencarian teks lengkap yang didukung Neptune. Lihat Pencarian teks lengkap Neptunus.
- **`Query hints`** – Blazegraph dan Neptune memperluas SPARQL menggunakan konsep petunjuk kueri. Selama migrasi, Anda perlu untuk memigrasi setiap petunjuk kueri yang Anda gunakan. Untuk informasi tentang dukungan Neptune petunjuk kueri terbaru, lihat Petunjuk kueri SPARQL.
- Inferensi – Blazegraph mendukung inferensi sebagai opsi yang dapat dikonfigurasi dalam mode tripel, tetapi tidak dalam mode quad. Neptune belum mendukung inferensi.
- Pencarian geospasial – Blazegraph mendukung konfigurasi ruang nama yang mengaktifkan dukungan geospasial. Fitur ini belum tersedia di Neptune.
- Multi-tenancy – Blazegraph mendukung multi-tenancy dalam basis data tunggal. Di Neptune, multi-tenancy didukung baik dengan menyimpan data dalam grafik bernama dan menggunakan

klausa USING NAMED untuk kueri SPARQL, atau dengan membuat klaster basis data terpisah untuk setiap tenant.

- Federasi — Neptunus saat ini mendukung federasi SPARQL 1.1 ke lokasi yang dapat diakses oleh instans Neptunus, seperti dalam VPC pribadi, di seberang, atau ke titik akhir internet eksternal. VPCs Tergantung pada pengaturan tertentu dan titik akhir federasi yang diperlukan, Anda mungkin memerlukan beberapa konfigurasi jaringan tambahan.
- Ekstensi standar Blazegraph – Blazegraph menyertakan beberapa ekstensi untuk standar SPARQL dan REST API, sedangkan Neptune hanya kompatibel dengan spesifikasi standar itu sendiri. Hal ini mungkin memerlukan perubahan pada aplikasi Anda, atau membuat migrasi menjadi sulit.

Penyediaan AWS infrastruktur untuk Neptunus

Meskipun Anda dapat membangun AWS infrastruktur yang diperlukan secara manual melalui Konsol Manajemen AWS atau AWS CLI, seringkali lebih nyaman untuk menggunakan CloudFormation template sebagai gantinya, seperti yang dijelaskan di bawah ini:

Penyediaan Neptunus menggunakan templat: CloudFormation

1. Navigasi ke Membuat cluster Amazon Neptunus menggunakan AWS CloudFormation.
2. Pilih Luncurkan Tumpukan di wilayah pilihan Anda.
3. Tetapkan parameter yang diperlukan (nama tumpukan dan `EC2SSHKeyPairName`). Juga tetapkan parameter pilihan berikut untuk memudahkan proses migrasi:
   - Atur `AttachBulkloadIAMRoleToNeptuneCluster` menjadi BETUL. Parameter ini memungkinkan untuk membuat dan melampirkan IAM role yang sesuai untuk klaster Anda sehingga mengizinkan bulk loading data.
   - Set `NotebookInstanceType` ke tipe instans pilihan Anda. Parameter ini membuat buku kerja Neptune yang Anda gunakan untuk menjalankan beban massal ke Neptune dan memvalidasi migrasi.
4. Pilih Berikutnya.
5. Mengatur pilihan tumpukan lain yang Anda inginkan.
6. Pilih Berikutnya.
7. Tinjau opsi Anda dan pilih kedua kotak centang untuk mengetahui bahwa AWS CloudFormation mungkin memerlukan kemampuan tambahan.

8. Pilih Membuat tumpukan.

Proses pembuatan tumpukan dapat memakan waktu beberapa menit.

Mengekspor data dari Blazegraph

Langkah selanjutnya adalah mengekspor data dari Blazegraph dalam format yang kompatibel dengan pemuat massal Neptune.

Tergantung pada bagaimana data disimpan dalam Blazegraph (tiga atau empat kali lipat) dan berapa banyak grafik bernama yang digunakan, Blazegraph mungkin mengharuskan Anda melakukan proses ekspor beberapa kali dan menghasilkan beberapa file data:

- Jika data disimpan sebagai tripel, Anda perlu menjalankan satu ekspor untuk setiap grafik bernama.
- Jika data disimpan sebagai quad, Anda dapat memilih untuk mengekspor data dalam format N-Quads atau mengekspor setiap grafik bernama dalam format tripel tiga kali lipat.

Di bawah ini kami berasumsi bahwa Anda mengekspor namespace tunggal sebagai N-Quads, tetapi Anda dapat mengulangi proses untuk ruang nama tambahan atau format ekspor yang diinginkan.

Jika Anda membutuhkan Blazegraph untuk online dan tersedia selama migrasi, gunakan kueri SPARQL CONSTRUCT. Hal ini mengharuskan Anda menginstal, mengkonfigurasi, dan menjalankan instans Blazegraph dengan titik akhir SPARQL yang dapat diakses.

Jika Anda tidak memerlukan Blazegraph untuk online, gunakan utilitas Ekspor. BlazeGraph Untuk melakukan ini, Anda harus men-download Blazegraph, dan file data dan file konfigurasi harus dapat diakses, tetapi server tidak perlu berjalan.

Mengekspor data dari Blazegraph menggunakan SPARQL CONSTRUCT

SPARQL CONSTRUCT adalah fitur dari SPARQL yang mengembalikan grafik RDF yang cocok dengan templat kueri yang ditentukan. Untuk kasus penggunaan ini, Anda menggunakannya untuk mengekspor data Anda satu namespace pada satu waktu, menggunakan kueri seperti berikut:

```
CONSTRUCT WHERE { hint:Query hint:analytic "true" . hint:Query
 hint:constructDistinctSPO "false" . ?s ?p ?o }
```

Meskipun alat RDF lain ada untuk mengekspor data ini, cara termudah untuk menjalankan kueri ini adalah dengan menggunakan titik akhir API REST yang disediakan oleh Blazegraph. Skrip berikut menunjukkan bagaimana menggunakan script Python (3.6+) untuk mengekspor data sebagai N-Quads:

```
import requests

# Configure the URL here: e.g. http://localhost:9999/sparql
url = "http://localhost:9999/sparql"
payload = {'query': 'CONSTRUCT WHERE { hint:Query hint:analytic "true" . hint:Query
 hint:constructDistinctSPO "false" . ?s ?p ?o }'}
# Set the export format to be n-quads
headers = {
'Accept': 'text/x-nquads'
}
# Run the http request
response = requests.request("POST", url, headers=headers, data = payload, files = [])
#open the file in write mode, write the results, and close the file handler
f = open("export.nq", "w")
f.write(response.text)
f.close()
```

Jika data disimpan sebagai tiga kali lipat, Anda perlu mengubah parameter `Accept` header untuk mengekspor data dalam format yang sesuai (N-Triples, RDF/XHTML, atau Turtle) menggunakan nilai yang ditentukan pada repo Blazegraph. GitHub

Menggunakan utilitas ekspor Blazegraph untuk mengekspor data

Blazegraph berisi metode utilitas untuk mengekspor data, yaitu kelas `ExportKB`. `ExportKB` memfasilitasi ekspor data dari Blazegraph, tetapi tidak seperti metode sebelumnya, mengharuskan server offline saat ekspor sedang berjalan. Ini menjadikannya metode ideal untuk digunakan ketika Anda dapat mengambil Blazegraph offline selama migrasi, atau migrasi dapat terjadi dari cadangan data.

Anda menjalankan utilitas dari baris perintah Java pada mesin yang memiliki Blazegraph diinstal tetapi tidak berjalan. Cara termudah untuk menjalankan perintah ini adalah dengan mengunduh rilis blazegraph.jar terbaru yang terletak di GitHub. Menjalankan perintah ini memerlukan beberapa parameter:

- **`log4j.primary.configuration`** – Lokasi file properti log4j.

- **`log4j.configuration`** – Lokasi file properti log4j.
- **`output`** – Direktori output untuk data yang diekspor. File terletak sebagai `tar.gz` dalam subdirektori bernama sebagai yang didokumentasikan dalam basis pengetahuan.
- **`format`** – Format output yang diinginkan diikuti oleh lokasi file ke `RWStore.properties`. Jika Anda bekerja dengan tripel, Anda perlu mengubah parameter `-format` ke `N-Triples`, `Turtle`, atau `RDF/XML`.

Misalnya, jika Anda memiliki file jurnal Blazegraph dan file properti, ekspor data sebagai N-Quads menggunakan kode berikut:

```
java -cp blazegraph.jar \
    com.bigdata.rdf.sail.ExportKB \
    -outdir ~/temp/ \
    -format N-Quads \
    ./RWStore.properties
```

Jika ekspor berhasil, Anda akan melihat output seperti ini:

```
Exporting kb as N-Quads on /home/ec2-user/temp/kb
Effective output directory: /home/ec2-user/temp/kb
Writing /home/ec2-user/temp/kb/kb.properties
Writing /home/ec2-user/temp/kb/data.nq.gz
Done
```

Buat bucket Amazon Simple Storage Service (Amazon S3) dan salin data yang diekspor ke dalamnya

Setelah Anda mengekspor data Anda dari Blazegraph, buat bucket Amazon Simple Storage Service (Amazon S3) di Wilayah yang sama dengan target klaster DB Neptune untuk loader massal Neptune gunakan untuk mengimpor data.

Untuk petunjuk tentang cara membuat bucket Amazon S3, lihat Bagaimana cara membuat Bucket S3? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon, dan Contoh pembuatan bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Untuk petunjuk tentang cara menyalin file data yang telah diekspor ke bucket Amazon S3 baru, lihat Mengunggah objek ke bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon,

atau Menggunakan perintah tingkat tinggi (s3) dengan CLI. AWS Anda juga dapat menggunakan kode Python seperti berikut untuk menyalin file satu per satu:

```
import boto3

region = 'region name'
bucket_name = 'bucket name'
s3 = boto3.resource('s3')
s3.meta.client.upload_file('export.nq', bucket_name, 'export.nq')
```

Gunakan pemuat massal Neptune untuk mengimpor data ke Neptune

Setelah mengekspor data Anda dari Blazegraph dan menyalinnya ke dalam bucket Amazon S3, Anda siap untuk mengimpor data ke Neptune. Neptune memiliki loader massal yang memuat data lebih cepat dan dengan sedikit overhead daripada melakukan operasi beban menggunakan SPARQL. Proses loader massal dimulai dengan panggilan ke loader titik akhir API untuk memuat data yang tersimpan dalam bucket S3 yang diidentifikasi ke Neptune.

Meskipun Anda bisa melakukan ini dengan panggilan langsung ke titik akhir loader REST, Anda harus memiliki akses ke VPC privat di mana instans Neptune target berjalan. Anda bisa mengatur bastion host, SSH ke mesin itu, dan menjalankan perintah cURL, tetapi menggunakan Neptune Workbench adalah lebih mudah.

Neptune Workbench adalah notebook Jupyter yang telah dikonfigurasi sebelumnya yang berjalan sebagai notebook SageMaker Amazon, dengan beberapa sihir notebook khusus Neptunus diinstal. Magics ini menyederhanakan operasi Neptune umum seperti memeriksa status klaster, menjalankan traversal SPARQL dan Gremlin, dan menjalankan operasi pemuatan massal.

Untuk memulai proses pemuatan massal menggunakan magic `%load`, yang menyediakan sebuah antarmuka untuk menjalankan Perintah Loader Neptune:

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Pilih aws-neptune-blazegraph-to-neptune.
3. Pilih Buka Notebook.
4. Di instans berjalan Jupyter, pilih notebook yang ada atau membuat notebook baru menggunakan kernel Python 3.
5. Dalam notebook Anda, buka sel, masukkan `%load`, dan jalankan sel.

6. Mengatur parameter untuk loader massal:

   a. Untuk Sumber, masukkan lokasi file sumber untuk diimpor: `s3://{bucket_name}/{file_name}`.

   b. Untuk Format, pilih format yang sesuai, yang dalam contoh ini `nquads`.

   c. Untuk Beban ARN, masukkan ARN untuk peran `IAMBulkLoad` (informasi ini terletak di konsol IAM di bawah Peran).

7. Pilih Kirim.

Hasilnya berisi status permintaan. Beban massal seringnya sebagai proses yang berjalan lama, sehingga respon tidak berarti bahwa beban telah selesai, hanya itu telah dimulai. Informasi status ini diperbarui secara berkala sampai melaporkan bahwa tugas selesai.

> **Note**
>
> Informasi ini juga tersedia di posting blog, Pindah ke cloud: Migrasi Blazegraph ke Amazon Neptune.

Memuat data ke Amazon Neptune

Ada beberapa cara berbeda untuk memuat data grafik ke Amazon Neptune:

- Jika Anda hanya perlu memuat jumlah data yang relatif kecil, Anda dapat menggunakan kueri seperti `INSERT` pernyataan SPARQL atau Gremlin `mergeV` dan `mergeE` langkah-langkah. openCypher juga menyertakan dan klausa. `CREATE` `MERGE`
- Anda dapat mengambil keuntungan dari Bulk Loader Neptune untuk menyerap sejumlah besar data yang berada di file eksternal. Perintah loader massal lebih cepat dan memiliki overhead kurang dari perintah bahasa kueri. Hal ini dioptimalkan untuk set data besar, dan mendukung baik data RDF (Resource Description Framework) maupun data Gremlin.
- Anda dapat menggunakan AWS Database Migration Service (AWS DMS) untuk mengimpor data dari penyimpanan data lain (lihatMenggunakan AWS Database Migration Service untuk memuat data ke Amazon Neptunus dari penyimpanan data yang berbeda, dan Panduan AWS Database Migration Service Pengguna).
- Untuk kumpulan data yang lebih kecil dalam satu atau beberapa file Amazon S3, Anda dapat menggunakan fungsi pemuatan berbasis kueri untuk membaca dan memproses data secara langsung dalam kueri Anda. Lihat Memuat data ke Amazon Neptunus menggunakan kueri untuk detail selengkapnya.

Topik

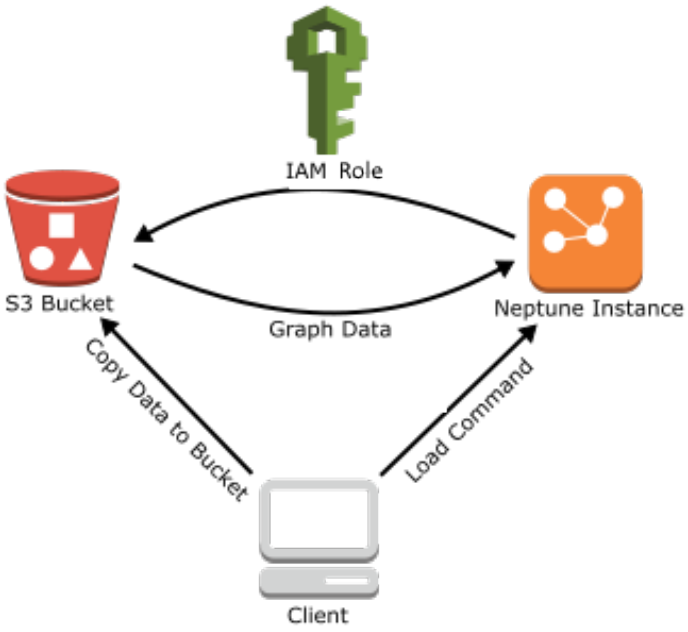
- Menggunakan pemuat massal Amazon Neptunus untuk menelan data
- Menggunakan AWS Database Migration Service untuk memuat data ke Amazon Neptunus dari penyimpanan data yang berbeda
- Memuat data ke Amazon Neptunus menggunakan kueri

Menggunakan pemuat massal Amazon Neptunus untuk menelan data

Amazon Neptunus menyediakan `Loader` perintah untuk memuat data dari file eksternal langsung ke cluster DB Neptunus. Anda dapat menggunakan perintah ini alih-alih mengeksekusi sejumlah besar pernyataan `INSERT`, langkah `addV` dan `addE`, atau panggilan API lainnya.

Perintah Loader Neptune lebih cepat, memiliki lebih sedikit overhead, dioptimalkan untuk dataset besar, dan mendukung data Gremlin dan data RDF (Resource Description Framework) yang digunakan oleh SPARQL.

Diagram berikut ini menunjukkan gambaran umum proses pemuatan.



Berikut adalah langkah-langkah proses pemuatan:

1. Salin file data ke bucket Amazon Simple Storage Service (Amazon S3).
2. Buat IAM role dengan akses Baca dan Daftar ke bucket.
3. Buat VPC Endpoint Amazon S3.
4. Mulai loader Neptune dengan mengirimkan permintaan melalui HTTP ke instans DB Neptune.
5. Instans DB Neptune mengasumsikan IAM role untuk memuat data dari bucket.

> **Note**
>
> Anda dapat memuat data terenkripsi dari Amazon S3 jika dienkripsi menggunakan Amazon `SSE-S3` S3 atau `SSE-KMS` mode, asalkan peran yang Anda gunakan untuk pemuatan massal memiliki akses ke objek Amazon S3, dan juga dalam kasus SSE-KMS, ke. `kms:decrypt` Neptunus kemudian dapat meniru kredensyal Anda dan mengeluarkan panggilan atas nama Anda. `s3:getObject`

Namun, Neptune saat ini tidak mendukung pemuatan data yang dienkripsi menggunakan Mode `SSE-C`.

Bagian berikut memberikan petunjuk untuk mempersiapkan dan memuat data ke Neptune.

Topik

- Prasyarat: IAM role dan Akses Amazon S3
- Muat Format Data
- Contoh: Memuat Data ke Instans DB Neptune
- Mengoptimalkan pemuatan massal Amazon Neptune
- Referensi Loader Neptune

Prasyarat: IAM role dan Akses Amazon S3

Memuat data dari bucket Amazon Simple Storage Service (Amazon S3) memerlukan AWS Identity and Access Management peran (IAM) yang memiliki akses ke bucket. Amazon Neptune mengasumsikan peran ini untuk memuat data.

> **Note**
>
> Anda dapat memuat data terenkripsi dari Amazon S3 jika dienkripsi menggunakan Mode `SSE-S3` Amazon S3. Dalam hal ini, Neptune dapat meniru kredensial Anda dan mengeluarkan panggilan `s3:getObject` atas nama Anda.
> Anda juga dapat memuat data terenkripsi dari Amazon S3 yang dienkripsi menggunakan `SSE-KMS`, selama IAM role Anda mencakup izin yang diperlukan untuk mengakses AWS KMS. Tanpa AWS KMS izin yang tepat, operasi beban massal gagal dan mengembalikan `LOAD_FAILED` respons.
> Neptune saat ini tidak mendukung pemuatan data yang dienkripsi Amazon S3 menggunakan Mode `SSE-C`.

Bagian berikut menunjukkan cara menggunakan kebijakan IAM terkelola untuk membuat peran IAM untuk mengakses sumber daya Amazon S3, lalu melampirkan peran tersebut ke cluster Neptunus Anda.

Topik

- Membuat peran IAM untuk memungkinkan Amazon Neptunus mengakses sumber daya Amazon S3
- Menambahkan IAM role ke Klaster Amazon Neptune
- Membuat VPC Endpoint Amazon S3
- Merantai peran IAM di Amazon Neptunus

> **Note**
>
> Petunjuk ini mengharuskan Anda memiliki akses ke konsol IAM dan izin untuk mengelola peran dan kebijakan IAM. Untuk informasi selengkapnya, lihat Izin untuk Bekerja di Konsol AWS Manajemen di Panduan Pengguna IAM.
> Konsol Amazon Neptune mengharuskan pengguna untuk memiliki izin IAM berikut untuk melampirkan peran ke klaster Neptune:
>
> ```
> iam:GetAccountSummary on resource: *
> iam:ListAccountAliases on resource: *
> iam:PassRole on resource: * with iam:PassedToService restricted to
>  rds.amazonaws.com
> ```

Membuat peran IAM untuk memungkinkan Amazon Neptunus mengakses sumber daya Amazon S3

Gunakan kebijakan IAM `AmazonS3ReadOnlyAccess` terkelola untuk membuat peran IAM baru yang memungkinkan Amazon Neptunus mengakses sumber daya Amazon S3.

Untuk membuat peran IAM baru yang memungkinkan akses Neptunus ke Amazon S3

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran.
3. Pilih Buat peran.
4. Pada layanan AWS , pilih S3.
5. Pilih Berikutnya: Izin.
6. Gunakan kotak filter untuk memfilter berdasarkan istilah S3 dan centang kotak di sebelah ReadOnlyAccessAmazonS3.

> **Note**
>
> Kebijakan ini memberikan izin `s3:Get*` dan `s3:List*` ke semua bucket. Langkah-langkah selanjutnya membatasi akses ke peran menggunakan kebijakan kepercayaan. Loader hanya membutuhkan izin `s3:Get*` dan `s3:List*` ke bucket tempat asal Anda memuat, sehingga Anda juga dapat membatasi izin ini dengan sumber daya Amazon S3. Jika bucket S3 Anda dienkripsi, Anda perlu menambahkan izin `kms:Decrypt`.

7. Pilih Berikutnya: Tinjauan.
8. Atur Nama Peran ke nama IAM role Anda, misalnya: `NeptuneLoadFromS3`. Anda juga dapat menambahkan nilai Deskripsi Peran opsional, seperti “Izinkan Neptune untuk mengakses sumber daya Amazon S3 atas nama Anda.”
9. Pilih Buat Peran.
10. Di panel navigasi, pilih Peran.
11. Di bidang Cari, masukkan nama peran yang Anda buat, dan pilih peran saat muncul di daftar.
12. Pilih tab Hubungan Kepercayaan, pilih Edit hubungan kepercayaan.
13. Di bidang teks, tempel kebijakan kepercayaan berikut.

    JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

14. Pilih Perbarui Kebijakan Kepercayaan.

15. Selesaikan langkah-langkah dalam Menambahkan IAM role ke Klaster Amazon Neptune.

Menambahkan IAM role ke Klaster Amazon Neptune

Gunakan konsol untuk menambahkan IAM role ke klaster Amazon Neptune. Hal ini memungkinkan setiap instans DB Neptune di klaster untuk mengasumsikan peran dan beban dari Amazon S3.

> **Note**
>
> Konsol Amazon Neptune mengharuskan pengguna untuk memiliki izin IAM berikut untuk melampirkan peran ke klaster Neptune:
>
> ```
> iam:GetAccountSummary on resource: *
> iam:ListAccountAliases on resource: *
> iam:PassRole on resource: * with iam:PassedToService restricted to
>  rds.amazonaws.com
> ```

Untuk menambahkan IAM role ke klaster Amazon Neptune

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Basis data.
3. Pilih pengenal cluster untuk cluster yang ingin Anda modifikasi.
4. Pilih tab Konektivitas & Keamanan.
5. Di bagian Peran IAM, pilih peran yang Anda buat di bagian sebelumnya.
6. Pilih Tambahkan peran.
7. Tunggu sampai IAM role menjadi dapat diakses ke klaster sebelum Anda menggunakannya.

Membuat VPC Endpoint Amazon S3

Pemuat Neptunus memerlukan titik akhir VPC tipe Gateway untuk Amazon S3.

Untuk menyiapkan akses untuk Amazon S3

1. Masuk ke Konsol Manajemen AWS dan buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc/

2. Di panel navigasi, pilih Titik Akhir.
3. Pilih Buat Titik Akhir.
4. Pilih Nama Layanan `com.amazonaws.region.s3` untuk titik akhir tipe Gateway.

> **Note**
>
> Jika Wilayah di sini salah, pastikan bahwa Wilayah konsol sudah benar.

5. Pilih VPC yang berisi instans DB Neptunus Anda (terdaftar untuk instans DB Anda di konsol Neptunus).
6. Pilih kotak centang di samping tabel rute yang terkait dengan subnet yang terkait dengan klaster Anda. Jika Anda hanya memiliki satu tabel rute, Anda harus memilih kotak itu.
7. Pilih Buat Titik Akhir.

Untuk informasi selengkapnya tentang membuat titik akhir, lihat VPC Endpoint dalam Panduan Pengguna Amazon VPC. Untuk informasi tentang keterbatasan VPC Endpoint, lihat VPC Endpoint untuk Amazon S3.

Langkah Berikutnya

Sekarang Anda telah memberikan akses ke bucket Amazon S3, Anda dapat mempersiapkan diri untuk memuat data. Untuk informasi tentang format yang didukung, lihat Muat Format Data.

Merantai peran IAM di Amazon Neptunus

> **Important**
>
> Fitur lintas akun beban massal baru yang diperkenalkan dalam rilis engine 1.2.1.0.R3 yang memanfaatkan peran IAM rantai dalam beberapa kasus dapat menyebabkan Anda mengamati kinerja beban curah yang menurun. Akibatnya, peningkatan ke rilis mesin yang mendukung fitur ini telah ditangguhkan sementara hingga masalah ini teratasi.

Saat Anda melampirkan peran ke klaster, klaster Anda dapat mengambil peran tersebut untuk mendapatkan akses ke data yang disimpan di Amazon S3. Dimulai dengan rilis mesin 1.2.1.0.R3, jika peran tersebut tidak memiliki akses ke semua sumber daya yang Anda butuhkan, Anda dapat merantai satu atau beberapa peran tambahan yang dapat diasumsikan oleh klaster Anda untuk

mendapatkan akses ke sumber daya lain. Setiap peran dalam rantai mengasumsikan peran berikutnya dalam rantai, sampai klaster Anda mengambil peran di akhir rantai.

Untuk peran rantai, Anda membangun hubungan kepercayaan di antara mereka. Misalnya, untuk berantai `RoleB``RoleA`, `RoleA` harus memiliki kebijakan izin yang memungkinkannya untuk berasumsi`RoleB`, dan `RoleB` harus memiliki kebijakan kepercayaan yang memungkinkannya meneruskan kembali izinnya. `RoleA` Untuk informasi selengkapnya, lihat Menggunakan peran IAM.

Peran pertama dalam rantai harus dilampirkan ke cluster yang memuat data.

Peran pertama, dan setiap peran berikutnya yang mengasumsikan peran berikut dalam rantai, harus memiliki:

- Kebijakan yang mencakup pernyataan spesifik dengan `Allow` efek pada `sts:AssumeRole` tindakan.
- Nama Sumber Daya Amazon (ARN) dari peran berikutnya dalam suatu `Resource` elemen.

> **Note**
>
> Bucket Amazon S3 target harus berada di AWS Wilayah yang sama dengan cluster.

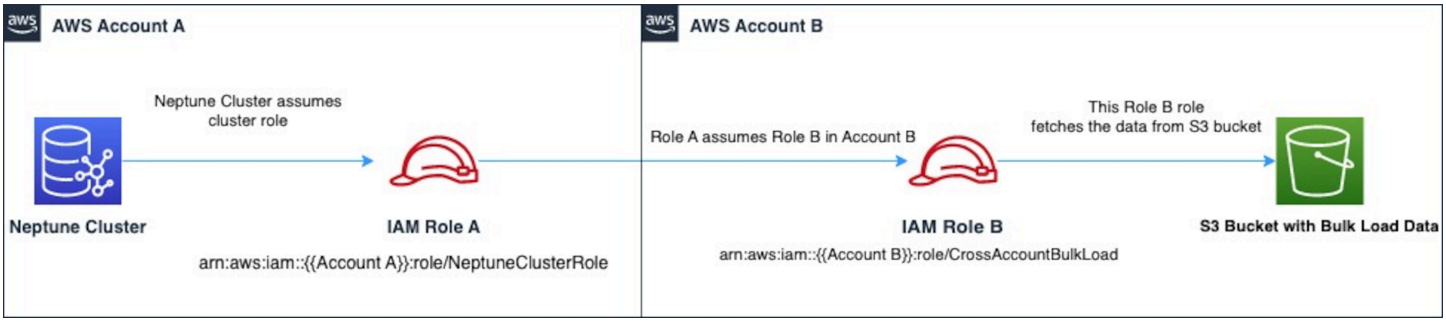
Akses lintas akun menggunakan peran berantai

Anda dapat memberikan akses lintas akun dengan merantai peran atau peran milik akun lain. Ketika klaster Anda sementara mengambil peran milik akun lain, klaster dapat memperoleh akses ke sumber daya di sana.

Misalnya, Akun A ingin mengakses data di bucket Amazon S3 milik Akun B:

- Akun A membuat peran AWS layanan untuk Neptunus `RoleA` bernama dan menempelkannya ke cluster.
- Akun B membuat peran bernama `RoleB` yang diizinkan untuk mengakses data dalam bucket Akun B.
- Akun A melampirkan kebijakan izin yang memungkinkannya untuk `RoleA` berasumsi. `RoleB`
- Akun B melampirkan kebijakan kepercayaan `RoleB` yang memungkinkannya meneruskan kembali izinnya. `RoleA`

- Untuk mengakses data di bucket Account B, Account A menjalankan perintah loader menggunakan `iamRoleArn` parameter yang berantai `RoleA` dan`RoleB`. Selama durasi operasi loader, `RoleA` maka untuk sementara mengasumsikan `RoleB` untuk mengakses bucket Amazon S3 di Akun B.



Misalnya, `RoleA` akan memiliki kebijakan kepercayaan yang membangun hubungan kepercayaan dengan Neptunus:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
          "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

`RoleA`juga akan memiliki kebijakan izin yang memungkinkannya untuk berasumsi`RoleB`, yang dimiliki oleh Akun B:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "Stmt1487639602000",
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/RoleB"
        }
    ]
}
```

Sebaliknya, `RoleB` akan memiliki kebijakan kepercayaan untuk membangun hubungan kepercayaan dengan`RoleA`:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/RoleA"
            }
        }
    ]
}
```

`RoleB`juga memerlukan izin untuk mengakses data di bucket Amazon S3 yang terletak di Akun B.

Membuat titik akhir VPC AWS Security Token Service (STS)

Pemuat Neptunus memerlukan titik akhir VPC saat Anda merantai peran IAM AWS STS untuk mengakses secara pribadi melalui alamat IP pribadi. AWS STS APIs Anda dapat terhubung langsung dari VPC Amazon ke AWS STS melalui VPC Endpoint VPC dengan cara yang aman dan terukur. Saat Anda menggunakan titik akhir VPC antarmuka, ini memberikan postur keamanan yang lebih baik karena Anda tidak perlu membuka firewall lalu lintas keluar. Ini juga memberikan manfaat lain menggunakan titik akhir Amazon VPC.

Saat menggunakan VPC Endpoint, lalu lintas ke AWS STS tidak mengirimkan melalui internet dan tidak pernah meninggalkan jaringan Amazon. VPC Anda terhubung dengan aman AWS STS tanpa risiko ketersediaan atau kendala bandwidth pada lalu lintas jaringan Anda. Untuk informasi selengkapnya, lihat Menggunakan titik AWS STS akhir VPC antarmuka.

Untuk mengatur akses untuk AWS Security Token Service (STS)

1. Masuk ke Konsol Manajemen AWS dan buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc/
2. Di panel navigasi, pilih Titik Akhir.
3. Pilih Buat Titik Akhir.
4. Pilih Nama Layanan: `com.amazonaws.region.sts` untuk titik akhir tipe Antarmuka.
5. Pilih VPC yang berisi instans DB Neptunus dan instans EC2 Anda.
6. Pilih kotak centang di sebelah subnet tempat instans EC2 Anda hadir. Anda tidak dapat memilih beberapa subnet dari Availability Zone yang sama.
7. Untuk jenis alamat IP, pilih dari opsi berikut:
   - IPv4— Tetapkan IPv4 alamat ke antarmuka jaringan titik akhir Anda. Opsi ini didukung hanya jika semua subnet yang dipilih memiliki rentang IPv4 alamat.
   - IPv6— Tetapkan IPv6 alamat ke antarmuka jaringan titik akhir Anda. Opsi ini didukung hanya jika semua subnet yang dipilih adalah subnet IPv6 -only.
   - Dualstack — Tetapkan keduanya IPv4 dan IPv6 alamat ke antarmuka jaringan endpoint Anda. Opsi ini didukung hanya jika semua subnet yang dipilih memiliki rentang keduanya IPv4 dan IPv6 alamat.
8. Untuk grup Keamanan, pilih grup keamanan yang akan diasosiasikan dengan antarmuka jaringan titik akhir untuk titik akhir VPC. Anda harus memilih semua grup keamanan yang dilampirkan ke instans DB Neptunus dan instans EC2 Anda.
9. Untuk Kebijakan, pilih Akses penuh untuk mengizinkan semua operasi oleh semua prinsipal di semua sumber daya melalui titik akhir VPC. Jika tidak, pilih Kustom untuk melampirkan kebijakan titik akhir VPC yang mengontrol izin yang dimiliki kepala sekolah untuk melakukan tindakan pada sumber daya melalui titik akhir VPC. Opsi ini hanya tersedia jika layanan mendukung kebijakan titik akhir VPC. Untuk informasi selengkapnya, lihat Kebijakan Endpoint.
10. (Opsional) Untuk menambahkan tag, pilih Tambahkan tag baru dan masukkan kunci tag dan nilai tag yang Anda inginkan.
11. Pilih Buat titik akhir.

Untuk informasi tentang membuat titik akhir, lihat Titik Akhir VPC di Panduan Pengguna Amazon VPC. Harap dicatat bahwa Amazon STS VPC Endpoint adalah prasyarat yang diperlukan untuk rantai peran IAM.

Sekarang setelah Anda memberikan akses ke AWS STS titik akhir, Anda dapat mempersiapkan untuk memuat data. Untuk informasi tentang format yang didukung, lihat Memuat Format Data.

Merantai peran dalam perintah loader

Anda dapat menentukan rantai peran saat menjalankan perintah loader dengan menyertakan daftar peran yang dipisahkan koma ARNs dalam parameter. `iamRoleArn`

Meskipun sebagian besar Anda hanya perlu memiliki dua peran dalam sebuah rantai, tentu saja mungkin untuk menyatukan tiga atau lebih. Misalnya, perintah loader ini merantai tiga peran:

```
curl -X POST https://localhost:8182/loader \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://(the target bucket name)/(the target date file name)",
        "iamRoleArn" : "arn:aws:iam::(Account A ID):role/(RoleA),arn:aws:iam::(Account
 B ID):role/(RoleB),arn:aws:iam::(Account C ID):role/(RoleC)",
        "format" : "csv",
        "region" : "us-east-1"
      }'
```

Muat Format Data

Amazon `Load` Neptunus API mendukung pemuatan data dalam berbagai format.

Format beban grafik properti

Data yang dimuat dalam salah satu format grafik properti berikut kemudian dapat ditanyakan menggunakan Gremlin dan OpenCypher:

- Format data muat Gremlin(`csv`): format nilai yang dipisahkan koma (CSV).
- Format beban data OpenCypher (`opencypher`): format nilai yang dipisahkan koma (CSV).

Format beban RDF

Untuk memuat data Resource Description Framework (RDF) yang Anda kueri menggunakan SPARQL, Anda dapat menggunakan salah satu format standar berikut seperti yang ditentukan oleh World Wide Web Consortium (W3C):

- N-Triples (`ntriples`) dari spesifikasi di. https://www.w3.org/TR/n-triples/
- N-Quads (`nquads`) dari spesifikasi di. https://www.w3.org/TR/n-quads/
- RDF/XML (`rdfxml`) dari spesifikasi di. https://www.w3.org/TR/rdf-syntax-grammar/
- Turtle (`turtle`) dari spesifikasi dihttps://www.w3.org/TR/turtle/.

Memuat data harus menggunakan pengkodean UTF-8

> **Important**
>
> Semua file load-data harus dikodekan dalam bentuk UTF-8. Jika file tidak dikodekan UTF-8, Neptunus mencoba memuatnya sebagai UTF-8.

Untuk data N-Quad dan N-Triple yang mencakup karakter Unicode, urutan escape `\u`*`xxxxx* didukung. Namun, Neptune tidak mendukung normalisasi. Jika ada nilai yang memerlukan normalisasi, itu tidak akan cocok byte-to-byte selama kueri. Untuk informasi selengkapnya tentang normalisasi, lihat halaman Normalisasi di Unicode.org.

Jika data Anda tidak dalam format yang didukung, Anda harus mengonversinya sebelum memuatnya.

Alat untuk mengonversi GraphMl ke format CSV Neptunus tersedia di proyek Graph CSV pada. ML2 GitHub

Dukungan kompresi untuk file load-data

Neptunus mendukung kompresi file `gzip` individual dalam atau format. `bzip2`

File terkompresi harus memiliki `.gz` atau `.bz2` ekstensi, dan harus berupa file teks tunggal yang dikodekan dalam format UTF-8. Anda dapat memuat banyak file, tetapi masing-masing harus berupa file teks terpisah`.gz`,`.bz2`, atau tidak terkompresi. Arsipkan file dengan ekstensi seperti `.tar.tar.gz`,, dan tidak `.tgz` didukung.

Bagian berikut menjelaskan format secara lebih detail.

Topik

- Format data muat Gremlin
- Muat format untuk data OpenCypher
- Format data beban RDF

Format data muat Gremlin

Untuk memuat data Apache TinkerPop Gremlin menggunakan format CSV, Anda harus menentukan simpul dan tepi dalam file terpisah.

Loader dapat memuat dari beberapa file vertex dan beberapa file edge dalam pekerjaan pemuatan tunggal.

Untuk setiap perintah pemuatan, rangkaian file yang akan dimuat harus dalam folder yang sama di bucket Amazon S3, dan Anda menentukan nama folder untuk parameter `source`. Nama file dan ekstensi nama file tidak penting.

Format CSV Amazon Neptune mengikuti spesifikasi dalam CSV RFC 4180. Untuk informasi selengkapnya, lihat Format Umum dan Tipe MIME untuk File CSV di situs web Internet Engineering Task Force (IETF).

> **Note**
>
> Semua file harus dienkode dalam format UTF-8.

Setiap file memiliki baris header yang dipisahkan koma. Baris header terdiri dari kedua header kolom sistem dan header kolom properti.

Header Kolom Sistem

Header kolom sistem yang diperlukan dan diizinkan berbeda untuk file vertex dan file edge.

Setiap kolom sistem hanya dapat muncul satu kali di header.

Semua label peka terhadap besar kecilnya huruf.

Header Vertex

- `~id`- Diperlukan

  ID untuk vertex.

- `~label`

  Label untuk vertex. Beberapa nilai label diperbolehkan, dipisahkan dengan titik koma (`;`).

  Jika tidak `~label` ada, TinkerPop berikan label dengan nilainya`vertex`, karena setiap simpul harus memiliki setidaknya satu label.

Header tepi

- `~id`- Diperlukan

  ID untuk edge.
- `~from`- Diperlukan

  ID vertex dari vertex from.
- `~to`- Diperlukan

  ID vertex dari vertex to.
- `~label`

  Label untuk edge. Edge hanya dapat memiliki satu label.

  Jika tidak `~label` ada, TinkerPop berikan label dengan nilainya`edge`, karena setiap tepi harus memiliki label.

Header Kolom Properti

Anda dapat menentukan kolom (`:`) untuk properti dengan menggunakan sintaks berikut. Nama jenis tidak peka dengan huruf besar/kecil. Perhatikan, bagaimanapun, bahwa jika titik dua muncul dalam nama properti, itu harus diloloskan dengan mendahuluinya dengan garis miring terbalik:. `\:`

```
propertyname:type
```

> **Note**
>
> Spasi, koma, carriage return, dan karakter baris baru tidak diperbolehkan di header kolom, sehingga nama properti tidak dapat menyertakan karakter ini.

Anda dapat menentukan kolom untuk jenis array dengan menambahkan `[]` ke jenisnya:

```
propertyname:type[]
```

> **Note**
>
> Properti edge hanya dapat memiliki satu nilai dan akan menyebabkan kesalahan jika jenis array yang ditentukan atau nilai kedua ditentukan.

Contoh berikut menunjukkan header kolom untuk properti bernama `age` dengan tipe `Int`.

```
age:Int
```

Setiap baris dalam file akan perlu memiliki integer dalam posisi itu atau dibiarkan kosong.

Array dari string diperbolehkan, tetapi string dalam array tidak dapat mencakup titik koma (`;`) kecuali ia di-escape menggunakan garis miring terbalik (seperti ini: `\;`).

Menentukan Kardinalitas Kolom

Header kolom dapat digunakan untuk menentukan kardinalitas untuk properti yang diidentifikasi oleh kolom. Hal ini memungkinkan loader massal untuk menghormati kardinalitas yang sama dengan cara yang kueri Gremlin lakukan.

Anda menentukan kardinalitas kolom seperti ini:

```
propertyname:type(cardinality)
```

*`cardinality`*Nilainya bisa berupa `single` atau`set`. Default-nya diasumsikan sebagai `set`, yang berarti bahwa kolom dapat menerima beberapa nilai. Dalam kasus file edge, kardinalitas selalu tunggal dan menentukan kardinalitas lainnya menyebabkan loader melempar pengecualian.

Jika kardinalitas `single`, loader melempar kesalahan jika nilai sebelumnya sudah ada ketika nilai dimuat, atau jika beberapa nilai dimuat. Perilaku ini dapat di-override sehingga nilai yang ada diganti ketika nilai baru dimuat dengan menggunakan bendera `updateSingleCardinalityProperties`. Lihat Perintah Loader.

Hal ini dimungkinkan untuk menggunakan pengaturan kardinalitas dengan tipe array, meskipun hal ini umumnya tidak diperlukan. Berikut adalah kombinasi yang mungkin:

- `name:type` — kardinalitasnya adalah `set`, dan kontennya bernilai tunggal.
- `name:type[]` — kardinalitasnya adalah `set`, dan kontennya multi-nilai.
- `name:type(single)` — kardinalitasnya adalah `single`, dan kontennya bernilai tunggal.
- `name:type(set)` — kardinalitasnya adalah `set`, yang sama dengan default, dan kontennya bernilai tunggal.
- `name:type(set)[]` — kardinalitasnya adalah `set`, dan kontennya multi-nilai.
- `name:type(single)[]` — ini bertentangan dan menyebabkan kesalahan dilempar.

Bagian berikut mencantumkan semua jenis data Gremlin yang tersedia.

Jenis Data Gremlin

Ini adalah daftar jenis properti yang diizinkan, dengan deskripsi masing-masing jenis.

Bool (atau Boolean)

Menunjukkan bidang Boolean. Nilai yang diizinkan: `false`, `true`

> **Note**
>
> Nilai apa pun selain `true` akan diperlakukan sebagai false.

Jenis Angka Utuh

Nilai-nilai di luar rentang yang didefinisikan menghasilkan kesalahan.

Tipe	Kisaran
Byte	-128 hingga 127
Pendek	-32768 ke 32767
Int	-2^31 hingga 2 ^ 31-1
Panjang	-2 ^ 63 hingga 2 ^ 63-1

Jenis Angka Desimal

Mendukung notasi desimal atau notasi ilmiah. Juga memungkinkan simbol seperti (+/-) Infinity atau NaN. INF tidak didukung.

Tipe	Kisaran
Desimal	Titik mengambang IEEE 754 32-bit
Ganda	Titik mengambang IEEE 754 64-bit

Nilai float dan double yang terlalu panjang dimuat dan dibulatkan ke nilai terdekat untuk presisi 24-bit (float) dan 53-bit (double). Sebuah nilai tengah dibulatkan ke 0 untuk digit terakhir yang tersisa di tingkat bit.

String

Tanda kutip adalah opsional. Koma, baris baru, dan karakter carriage return secara otomatis di-escape jika mereka termasuk dalam string yang diapit oleh tanda kutip ganda (`"`). Contoh: `"Hello, World"`

Untuk memasukkan tanda kutip dalam string bertanda kutip, Anda dapat meng-escape tanda kutip dengan menggunakan dua berturut-turut: Contoh: `"Hello ""World"""`

Array dari string diperbolehkan, tetapi string dalam array tidak dapat mencakup titik koma (`;`) kecuali ia di-escape menggunakan garis miring terbalik (seperti ini: `\;`).

Jika Anda ingin mengapit string dalam array dengan tanda kutip, Anda harus mengapit keseluruhan array dengan satu set tanda kutip. Contoh: `"String one; String 2; String 3"`

Date

Tanggal Java dalam format ISO-8601. Mendukung format berikut:`yyyy-MM-dd`,`yyyy-MM-ddTHH:mm`,`yyyy-MM-ddTHH:mm:ss`,`yyyy-MM-ddTHH:mm:ssZ`. Nilai-nilai dikonversi ke waktu epoch dan disimpan.

Datetime

Tanggal Java dalam format ISO-8601. Mendukung format berikut:`yyyy-MM-dd`,`yyyy-MM-ddTHH:mm`,`yyyy-MM-ddTHH:mm:ss`,`yyyy-MM-ddTHH:mm:ssZ`. Nilai-nilai dikonversi ke waktu epoch dan disimpan.

Format Baris Gremlin

Pembatas

Bidang dalam baris dipisahkan dengan koma. Catatan dipisahkan oleh baris baru atau baris baru yang diikuti dengan carriage return.

Bidang Kosong

Bidang kosong diperbolehkan untuk kolom yang tidak diperlukan (seperti properti yang ditetapkan pengguna). Bidang kosong masih memerlukan pemisah koma. Bidang kosong pada kolom yang diperlukan akan menghasilkan kesalahan penguraian. Nilai string kosong ditafsirkan sebagai nilai string kosong untuk bidang; bukan sebagai bidang kosong. Contoh pada bagian berikutnya memiliki bidang kosong di setiap vertex contoh.

Vertex IDs

Nilai `~id` harus unik untuk semua vertex di setiap file vertex. Beberapa baris vertex dengan nilai-nilai `~id` identik diterapkan ke satu vertex dalam grafik. String kosong (`""`) adalah id yang valid, dan simpul dibuat dengan string kosong sebagai id.

Tepi IDs

Selain itu, nilai `~id` harus unik untuk semua edge di setiap file edge. Beberapa baris edge dengan nilai-nilai `~id` identik diterapkan ke satu edge dalam grafik. String kosong (`""`) adalah id yang valid, dan tepi dibuat dengan string kosong sebagai id.

Label

Label peka huruf besar/kecil dan tidak boleh kosong. Nilai `""` akan menghasilkan kesalahan.

Nilai String

Tanda kutip adalah opsional. Koma, baris baru, dan karakter carriage return secara otomatis di-escape jika mereka termasuk dalam string yang diapit oleh tanda kutip ganda (`"`). Nilai string kosong (`""`) ditafsirkan sebagai nilai string kosong untuk bidang; bukan sebagai bidang kosong.

Spesifikasi Format CSV

Format CSV Neptune mengikuti spesifikasi CSV RFC 4180, termasuk persyaratan berikut ini.

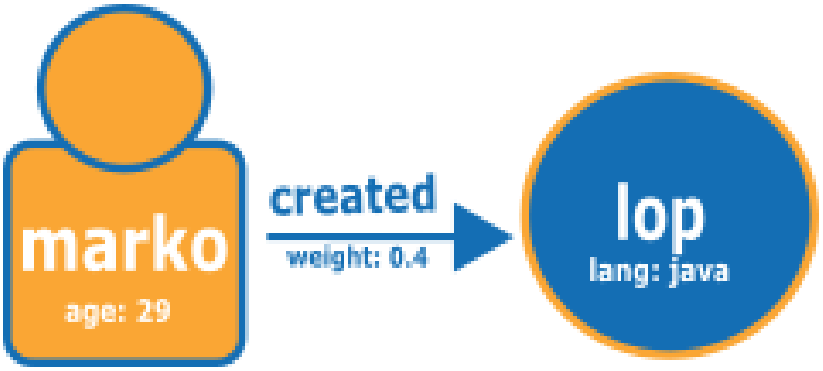
- Akhir baris gaya Unix dan Windows didukung (\n atau \r n).

- Setiap bidang dapat dikutip (menggunakan tanda kutip ganda).
- Bidang yang berisi line-break, double-quote, atau koma harus dikutip. (Jika tidak, pemuatan dibatalkan segera.)
- Sebuah karakter tanda kutip ganda (") dalam bidang harus diwakili oleh dua (dua) karakter tanda kutip. Misalnya, string `Hello "World"` harus ada sebagai `"Hello ""World"""` dalam data.
- Ruang sekitar antara delimiter diabaikan. Jika baris hadir sebagai`value1, value2`, mereka disimpan sebagai `"value1"` dan`"value2"`.
- Setiap karakter lainnya yang di-escape disimpan verbatim. Misalnya, `"data1\tdata2"` disimpan sebagai `"data1\tdata2"`. Tidak ada proses escape lebih lanjut yang diperlukan selama karakter ini diapit dalam tanda kutip.
- Bidang kosong diperbolehkan. Bidang kosong dianggap sebagai nilai kosong.
- Beberapa nilai untuk satu bidang ditentukan dengan titik koma (`;`) antara nilai-nilai.

Untuk informasi selengkapnya, lihat Format Umum dan Tipe MIME untuk File CSV di situs web Internet Engineering Task Force (IETF).

Contoh Gremlin

Diagram berikut menunjukkan contoh dua simpul dan tepi yang diambil dari Grafik TinkerPop Modern.



Berikut ini adalah grafik dalam format pemuatan CSV Neptune.

File vertex:

```
~id,name:String,age:Int,lang:String,interests:String[],~label
v1,"marko",29,,"sailing;graphs",person
v2,"lop",,"java",,software
```

Tampilan tabular dari file vertex:

~id	Nama:string	Umur: int	lang:string	Minat:string []	~label
v1	“marko”	29		["berlayar”, “grafik"]	pribadi
v2	“lop”		“jawa”		software

File edge:

```
~id,~from,~to,~label,weight:Double
e1,v1,v2,created,0.4
```

Tampilan tabular dari file edge:

~id	~dari	~untuk	~label	Berat: ganda
e1	v1	v2	dibuat	0,4

Langkah Berikutnya

Sekarang setelah Anda tahu lebih banyak tentang format pemuatan, lihat Contoh: Memuat Data ke Instans DB Neptune.

Muat format untuk data OpenCypher

Untuk memuat data OpenCypher menggunakan format CSV OpenCypher, Anda harus menentukan node dan hubungan dalam file terpisah. Loader dapat memuat dari beberapa file node dan file hubungan ini dalam satu tugas pemuatan.

Untuk setiap perintah pemuatan, kumpulan file yang akan dimuat harus memiliki awalan jalur yang sama di bucket Amazon Simple Storage Service. Anda menentukan awalan itu di parameter sumber. Nama file dan ekstensi sebenarnya tidak penting.

Di Amazon Neptunus, format CSV OpenCypher sesuai dengan spesifikasi CSV RFC 4180. Untuk informasi selengkapnya, lihat Format Umum dan Jenis MIME untuk File CSV (https://tools.ietf.org/html/rfc4180) di situs web Internet Engineering Task Force (IETF).

> **Note**
>
> File-file ini HARUS dikodekan dalam format UTF-8.

Setiap file memiliki baris header yang dipisahkan koma yang berisi header kolom sistem dan header kolom properti.

Header kolom sistem dalam file pemuatan data OpenCypher

Kolom sistem yang diberikan hanya dapat muncul sekali di setiap file. Semua label header kolom sistem peka huruf besar/kecil.

Header kolom sistem yang diperlukan dan diizinkan berbeda untuk file pemuatan node OpenCypher dan file pemuatan hubungan:

Header kolom sistem dalam file node

- **`:ID`**— (Wajib) ID untuk node.

  Ruang ID opsional dapat ditambahkan ke header `:ID` kolom node seperti ini:`:ID(`*`ID Space`*`)`. Contohnya adalah `:ID(movies)`.

  Saat memuat hubungan yang menghubungkan node dalam file ini, gunakan spasi ID yang sama di `:START_ID` and/or `:END_ID` kolom file hubungan.

  `:ID`Kolom node opsional dapat disimpan sebagai properti dalam bentuk,*`property name`*`:ID`. Contohnya adalah `name:ID`.

  Node IDs harus unik di semua file node dalam beban saat ini dan sebelumnya. Jika ruang ID digunakan, node IDs harus unik di semua file node yang menggunakan ruang ID yang sama dalam beban saat ini dan sebelumnya.

- **`:LABEL`**— Label untuk node.

  Saat menggunakan beberapa nilai label untuk satu node, setiap label harus dipisahkan dengan titik koma (). `;`

Header kolom sistem dalam file hubungan

- **`:ID`**— ID untuk hubungan. Ini diperlukan ketika `userProvidedEdgeIds` benar (default), tetapi tidak valid kapan `userProvidedEdgeIds` benar. `false`

  Hubungan IDs harus unik di semua file hubungan dalam pemuatan saat ini dan sebelumnya.

- **`:START_ID`**— (Wajib) ID node dari node hubungan ini dimulai dari.

  Secara opsional, ruang ID dapat dikaitkan dengan kolom ID awal dalam formulir`:START_ID(`*`ID Space`*`)`. Ruang ID yang ditetapkan ke ID node awal harus cocok dengan ruang ID yang ditetapkan ke node dalam file node.

- **`:END_ID`**— (Wajib) ID node dari node hubungan ini berakhir di.

  Secara opsional, ruang ID dapat dikaitkan dengan kolom ID akhir dalam formulir`:END_ID(`*`ID Space`*`)`. Ruang ID yang ditetapkan ke ID node akhir harus cocok dengan ruang ID yang ditetapkan ke node dalam file simpulnya.

- **`:TYPE`**— Tipe untuk hubungan. Hubungan hanya dapat memiliki satu jenis.

> **Note**
>
> Lihat Memuat data OpenCypher untuk informasi tentang bagaimana duplikat node atau relasi IDs ditangani oleh proses pemuatan massal.

Header kolom properti di file pemuatan data OpenCypher

Anda dapat menentukan bahwa kolom menyimpan nilai untuk properti tertentu menggunakan header kolom properti dalam bentuk berikut:

```
propertyname:type
```

Spasi, koma, carriage return, dan karakter baris baru tidak diperbolehkan di header kolom, sehingga nama properti tidak dapat menyertakan karakter ini. Berikut adalah contoh header kolom untuk properti bernama `age` tipe`Int`:

```
age:Int
```

Kolom dengan `age:Int` sebagai header kolom kemudian harus berisi integer atau nilai kosong di setiap baris.

Tipe data dalam file pemuatan data Neptunus OpenCypher

- **`Bool`**atau **`Boolean`**— Sebuah bidang Boolean. Nilai yang diizinkan adalah `true` dan `false`.

  Nilai apa pun selain `true` diperlakukan sebagai`false`.
- **`Byte`**— Sebuah bilangan bulat dalam rentang `-128` melalui`127`.
- **`Short`**— Sebuah bilangan bulat dalam rentang `-32,768` melalui`32,767`.
- **`Int`**— Sebuah bilangan bulat dalam rentang `-2^31` melalui`2^31 - 1`.
- **`Long`**— Sebuah bilangan bulat dalam rentang `-2^63` melalui`2^63 - 1`.
- **`Float`**— Nomor floating point IEEE 754 32-bit. Notasi desimal dan notasi ilmiah keduanya didukung. `Infinity`,`-Infinity`, dan `NaN` semuanya diakui, tetapi `INF` tidak.

  Nilai dengan terlalu banyak digit yang cocok dibulatkan ke nilai terdekat (nilai tengah dibulatkan ke 0 untuk digit terakhir yang tersisa di tingkat bit).
- **`Double`**— Nomor floating point IEEE 754 64-bit. Notasi desimal dan notasi ilmiah keduanya didukung. `Infinity`,`-Infinity`, dan `NaN` semuanya diakui, tetapi `INF` tidak.

  Nilai dengan terlalu banyak digit yang cocok dibulatkan ke nilai terdekat (nilai tengah dibulatkan ke 0 untuk digit terakhir yang tersisa di tingkat bit).
- **`String`**- Tanda kutip adalah opsional. Karakter comma, newline, dan carriage return secara otomatis lolos jika disertakan dalam string yang dikelilingi oleh tanda kutip ganda () seperti. `"` `"Hello, World"`

  Anda dapat menyertakan tanda kutip dalam string yang dikutip dengan menggunakan dua berturut-turut, seperti. `"Hello ""World"""`
- **`DateTime`**— Tanggal Java dalam salah satu format ISO-8601 berikut:
  - `yyyy-MM-dd`
  - `yyyy-MM-ddTHH:mm`
  - `yyyy-MM-ddTHH:mm:ss`
  - `yyyy-MM-ddTHH:mm:ssZ`

Jenis data cast otomatis dalam file pemuatan data Neptunus OpenCypher

Tipe data auto-cast disediakan untuk memuat tipe data yang saat ini tidak didukung secara native oleh Neptunus. Data dalam kolom tersebut disimpan sebagai string, kata demi kata tanpa verifikasi terhadap format yang dimaksudkan. Tipe data auto-cast berikut diperbolehkan:

- **`Char`**— Sebuah `Char` bidang. Disimpan sebagai string.
- **`Date`**, **`LocalDate`**, dan **`LocalDateTime`**, — Lihat Neo4j Instans Temporal untuk deskripsi`date`,`localdate`, dan jenis. `localdatetime` Nilai-nilai dimuat kata demi kata sebagai string, tanpa validasi.
- **`Duration`**— Lihat format Durasi Neo4j. Nilai-nilai dimuat kata demi kata sebagai string, tanpa validasi.
- Titik — Bidang titik, untuk menyimpan data spasial. Lihat Instan spasial. Nilai-nilai dimuat kata demi kata sebagai string, tanpa validasi.

Contoh format beban OpenCypher

Diagram berikut yang diambil dari Grafik TinkerPop Modern menunjukkan contoh dua node dan hubungan:

Berikut ini adalah grafik dalam format beban Neptunus OpenCypher normal.

Berkas simpul:

```
:ID,name:String,age:Int,lang:String,:LABEL
v1,"marko",29,,person
v2,"lop",,"java",software
```

File hubungan:

```
:ID,:START_ID,:END_ID,:TYPE,weight:Double
e1,v1,v2,created,0.4
```

Atau, Anda dapat menggunakan spasi ID dan ID sebagai properti, sebagai berikut:

File simpul pertama:

```
name:ID(person),age:Int,lang:String,:LABEL
"marko",29,,person
```

File simpul kedua:

```
name:ID(software),age:Int,lang:String,:LABEL
"lop",,"java",software
```

File hubungan:

```
:ID,:START_ID(person),:END_ID(software),:TYPE,weight:Double
e1,"marko","lop",created,0.4
```

Format data beban RDF

Untuk memuat data Resource Description Framework (RDF), Anda dapat menggunakan salah satu format standar berikut seperti yang ditentukan oleh World Wide Web Consortium (W3C):

- N-Triples (`ntriples`) dari spesifikasi di https://www.w3.org/TR/n-triples/
- N -Quad (`nquads`) dari spesifikasinya di https://www.w3.org/TR/n-quads/
- RDF/XML (`rdfxml`) dari spesifikasinya di https://www.w3.org/TR/rdf-syntax-grammar/
- Turtle (`turtle`) dari spesifikasi di https://www.w3.org/TR/turtle/

> **Important**
>
> Semua file harus dienkode dalam format UTF-8.
> Untuk data N-Quad dan N-Triple yang mencakup karakter Unicode, urutan escape \u*xxxxx* didukung. Namun, Neptune tidak mendukung normalisasi. Jika ada nilai yang memerlukan normalisasi, itu tidak akan cocok byte-to-byte selama kueri. Untuk informasi selengkapnya tentang normalisasi, lihat halaman Normalisasi di Unicode.org.

Langkah Berikutnya

Sekarang setelah Anda tahu lebih banyak tentang format pemuatan, lihat Contoh: Memuat Data ke Instans DB Neptune.

Contoh: Memuat Data ke Instans DB Neptune

Contoh ini menunjukkan cara memuat data ke Amazon Neptune. Kecuali dinyatakan lain, Anda harus mengikuti langkah-langkah dari instans Amazon Elastic Compute Cloud (Amazon EC2) di Amazon Virtual Private Cloud (VPC) yang sama dengan instans DB Neptune Anda.

Prasyarat untuk Contoh Pemuatan Data

Sebelum memulai, Anda harus memiliki hal-hal berikut:

- Instans DB Neptune.

  Untuk informasi tentang meluncurkan instans DB Neptune, lihat Membuat cluster Amazon Neptunus.
- Bucket Amazon Simple Storage Service (Amazon S3) tempat file data akan diletakkan.

  Anda dapat menggunakan bucket yang ada. Jika Anda tidak memiliki bucket S3, lihat Buat Bucket di Panduan Memulai Amazon S3.
- Grafik data untuk dimuat, dalam salah satu format yang didukung oleh loader Neptune:

  Jika Anda menggunakan Gremlin untuk menanyakan grafik Anda, Neptunus dapat memuat data dalam format comma-separated-values (CSV), seperti yang dijelaskan dalam. Format data muat Gremlin

  Jika Anda menggunakan OpenCypher untuk menanyakan grafik Anda, Neptunus juga dapat memuat data dalam format khusus OpenCypher, seperti yang dijelaskan dalam. CSV Muat format untuk data OpenCypher

  Jika Anda menggunakan SPARQL, Neptune dapat memuat data dalam sejumlah format RDF, seperti yang dijelaskan di Format data beban RDF.
- IAM role untuk instans DB Neptune menganggap bahwa memiliki kebijakan IAM yang memungkinkan akses ke file data dalam bucket S3. Kebijakan harus memberikan izin Baca dan Daftar.

Untuk informasi tentang membuat peran yang memiliki akses ke Amazon S3 dan kemudian mengaitkannya dengan klaster Neptune, lihat Prasyarat: IAM role dan Akses Amazon S3.

> **Note**
>
> API `Load` Neptune membutuhkan akses baca ke file data saja. Kebijakan IAM tidak perlu mengizinkan akses tulis atau akses ke seluruh bucket.

- VPC Endpoint Amazon S3. Untuk informasi selengkapnya, lihat bagian Membuat VPC Endpoint Amazon S3.

Membuat VPC Endpoint Amazon S3

Neptune loader membutuhkan VPC Endpoint untuk Amazon S3.

Untuk menyiapkan akses untuk Amazon S3

1. Masuk ke Konsol Manajemen AWS dan buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc/
2. Pada panel navigasi kiri, pilih Titik Akhir.
3. Pilih Buat Titik Akhir.
4. Pilih Nama Layanan `com.amazonaws.`*`region`*`.s3`.

   > **Note**
   >
   > Jika Wilayah di sini salah, pastikan bahwa Wilayah konsol sudah benar.

5. Pilih VPC yang berisi instans DB Neptune Anda.
6. Pilih kotak centang di samping tabel rute yang terkait dengan subnet yang terkait dengan klaster Anda. Jika Anda hanya memiliki satu tabel rute, Anda harus memilih kotak itu.
7. Pilih Buat Titik Akhir.

Untuk informasi selengkapnya tentang membuat titik akhir, lihat VPC Endpoint dalam Panduan Pengguna Amazon VPC. Untuk informasi tentang keterbatasan VPC Endpoint, lihat VPC Endpoint untuk Amazon S3.

Untuk memuat data ke dalam instans DB Neptune

1. Salin file data ke bucket Amazon S3. Bucket S3 harus berada di AWS Region yang sama dengan cluster yang memuat data.

   Anda dapat menggunakan AWS CLI perintah berikut untuk menyalin file ke ember.

   > **Note**
   >
   > Perintah ini tidak perlu dijalankan dari instans Amazon EC2.

   ```
   aws s3 cp data-file-name s3://bucket-name/object-key-name
   ```

   > **Note**
   >
   > Di Amazon S3, sebuah nama kunci objek adalah seluruh jalur file, termasuk nama file. Contoh: Dalam perintah `aws s3 cp datafile.txt s3://examplebucket/mydirectory/datafile.txt`, nama kunci objeknya adalah **`mydirectory/datafile.txt`**.

   Atau, Anda dapat menggunakan file Konsol Manajemen AWS untuk mengunggah file ke bucket S3. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/, dan pilih ember. Di sudut kiri atas, pilih Unggah untuk mengunggah file.

2. Dari jendela baris perintah, masukkan yang berikut ini untuk menjalankan pemuat Neptunus, menggunakan nilai yang benar untuk titik akhir, jalur Amazon S3, format, dan peran IAM ARN.

   `format`Parameter dapat berupa salah satu dari nilai berikut: `csv` untuk Gremlin, `opencypher` untuk OpenCypher, atau`ntriples`,,, dan untuk RDF. `nquads` `turtle` `rdfxml` Untuk informasi tentang parameter lain, lihat Perintah Loader Neptune.

   Untuk informasi tentang menemukan nama host instans DB Neptune Anda, lihat bagian Menghubungkan ke Titik Akhir Amazon Neptune..

   Parameter Wilayah harus sesuai dengan Wilayah klaster dan bucket S3.

   Amazon Neptunus tersedia di Wilayah berikut: AWS

- US East (N. Virginia): `us-east-1`
- AS Timur (Ohio): `us-east-2`
- US West (N. California): `us-west-1`
- US West (Oregon): `us-west-2`
- Canada (Central): `ca-central-1`
- Kanada Barat (Calgary): `ca-west-1`
- South America (São Paulo): `sa-east-1`
- Eropa (Stockholm): `eu-north-1`
- Eropa (Spanyol): `eu-south-2`
- Eropa (Irlandia): `eu-west-1`
- Eropa (London): `eu-west-2`
- Eropa (Paris): `eu-west-3`
- Eropa (Frankfurt): `eu-central-1`
- Timur Tengah (Bahrain): `me-south-1`
- Timur Tengah (UEA): `me-central-1`
- Israel (Tel Aviv): `il-central-1`
- Afrika (Cape Town): `af-south-1`
- Asia Pasifik (Hong Kong): `ap-east-1`
- Asia Pacific (Tokyo): `ap-northeast-1`
- Asia Pasifik (Seoul): `ap-northeast-2`
- Asia Pasifik (Osaka): `ap-northeast-3`
- Asia Pacific (Singapore): `ap-southeast-1`
- Asia Pacific (Sydney): `ap-southeast-2`
- Asia Pasifik (Jakarta): `ap-southeast-3`
- Asia Pasifik (Melbourne): `ap-southeast-4`
- Asia Pasifik (Malaysia): `ap-southeast-5`
- Asia Pasifik (Mumbai): `ap-south-1`
- Asia Pasifik (Hyderabad): `ap-south-2`
- Tiongkok (Beijing): `cn-north-1`
- Tiongkok (Ningxia): `cn-northwest-1`

- AWS GovCloud (AS-Barat): `us-gov-west-1`
- AWS GovCloud (AS-Timur): `us-gov-east-1`

```
curl -X POST \
    -H 'Content-Type: application/json' \
    https://your-neptune-endpoint:port/loader -d '
    {
      "source" : "s3://bucket-name/object-key-name",
      "format" : "format",
      "iamRoleArn" : "arn:aws:iam::account-id:role/role-name",
      "region" : "region",
      "failOnError" : "FALSE",
      "parallelism" : "MEDIUM",
      "updateSingleCardinalityProperties" : "FALSE",
      "queueRequest" : "TRUE",
      "dependencies" : ["load_A_id", "load_B_id"]
    }'
```

Untuk informasi tentang membuat dan mengaitkan IAM role dengan klaster Neptune, lihat Prasyarat: IAM role dan Akses Amazon S3.

> **Note**
>
> Lihat Parameter Permintaan Loader Neptune) untuk informasi detail tentang parameter permintaan pemuatan. Singkatnya:
> Parameter `source` menerima URI Amazon S3 yang menunjuk ke satu file atau folder.
> Jika Anda menentukan folder, Neptune memuat setiap file data dalam folder.
> Folder dapat berisi beberapa file vertex dan beberapa file edge.
> URI dapat berupa format berikut.
>
> - `s3://bucket_name/object-key-name`
> - `https://s3.amazonaws.com/bucket_name/object-key-name`
> - `https://s3-us-east-1.amazonaws.com/bucket_name/object-key-name`
>
> Parameter `format` dapat berupa salah satu dari hal berikut:
>
> - Format CSV Gremlin (`csv`) untuk grafik properti Gremlin
> - OpenCypher CSV format (`opencypher`) untuk grafik properti OpenCypher

- Format N -Triple (`ntriples`) untuk RDF/SPARQL
- Format N-Quad (`nquads`) untuk RDF/SPARQL
- Format RDF/XML (`rdfxml`) untuk RDF/SPARQL
- Format Turtle (`turtle`) untuk RDF/SPARQL

Parameter `parallelism` opsional memungkinkan Anda membatasi jumlah utas yang digunakan dalam proses pemuatan massal. Anda dapat mengaturnya ke `LOW`, `MEDIUM`, `HIGH`, atau `OVERSUBSCRIBE`.
Saat `updateSingleCardinalityProperties` diatur ke `"FALSE"`, loader mengembalikan kesalahan jika lebih dari satu nilai disediakan dalam file sumber yang dimuat untuk edge atau properti single-cardinality vertex.
Mengatur `queueRequest` ke `"TRUE"` menyebabkan permintaan pemuatan ditempatkan dalam antrean jika sudah ada pekerjaan pemuatan yang berjalan.
Parameter `dependencies` membuat eksekusi permintaan pemuatan dapat berubah pada penyelesaian satu pekerjaa pemuatan atau lebih yang berhasil yang telah ditempatkan dalam antrean.

3. Neptune loader mengembalikan pekerjaan `id` yang memungkinkan Anda untuk memeriksa status atau membatalkan proses pemuatan; misalnya:

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

4. Masukkan hal berikut ini untuk mendapatkan status pemuatan dengan `loadId` dari Langkah 3:

```
curl -G 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5'
```

Jika status pemuatan mencantumkan kesalahan, Anda dapat meminta status yang lebih rinci dan daftar kesalahan. Untuk informasi selengkapnya dan contoh tambahan, lihat Neptunus Loader API Get-Status.

5. (Opsional) Membatalkan pekerjaan `Load`.

Masukkan hal berikut untuk `Delete` pekerjaan loader dengan pekerjaan `id` dari Langkah 3:

```
curl -X DELETE 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5'
```

Perintah `DELETE` mengembalikan kode HTTP `200 OK` setelah pembatalan berhasil.

Data dari file dari pekerjaan pemuatan yang telah selesai melakukan pemuatan tidak di-rollback. Data tetap ada dalam instans DB Neptune.

Mengoptimalkan pemuatan massal Amazon Neptune

Gunakan strategi berikut untuk menjaga waktu muat hingga minimum untuk pemuatan massal Neptune:

- Bersihkan data Anda:
  - Pastikan untuk mengonversi data Anda menjadi Format data yang didukung sebelum memuat.
  - Hapus duplikat atau kesalahan yang diketahui.
  - Kurangi jumlah predikat unik (seperti properti edge dan vertex) sebanyak yang Anda bisa.
- Optimalkan file Anda:
  - Jika Anda memuat file besar seperti file CSV dari bucket Amazon S3, loader mengelola konkurensi untuk Anda dengan menguraikannya menjadi potongan-potongan yang dapat dimuat secara paralel. Menggunakan sejumlah besar file kecil dapat memperlambat proses ini.
  - Jika Anda memuat beberapa file dari awalan Amazon S3, loader secara otomatis memuat file vertex terlebih dahulu, lalu file edge sesudahnya. Namun, jika Anda tahu bahwa Anda hanya akan memuat file tepi, `edgeOnlyLoad` dapat diatur `TRUE` untuk melewati lintasan pertama di mana semua file dipindai untuk menentukan isinya (simpul atau tepi) sehingga file simpul apa pun yang ditemukan dimuat sebelum file tepi. Ini dapat mempercepat waktu buka secara signifikan, terutama ketika banyak file edge terlibat. Jika beberapa file simpul juga ada dalam awalan `source` (parameter) Amazon S3 yang sama, mereka akan dimuat tetapi tanpa jaminan pemesanan apa pun relatif terhadap file lain. Juga, jika beberapa `from` atau `to` simpul tidak ada dalam database, penyisipan tepi dapat melaporkan kesalahan dengan pesan. `FROM_OR_TO_VERTEX_ARE_MISSING` Sebagai praktik terbaik, letakkan node dan tepi di awalan Amazon S3 yang terpisah.
- Periksa pengaturan loader Anda:

- Jika Anda tidak perlu melakukan operasi lain selama pemuatan, gunakan `OVERSUBSCRIBE` `parallelism`parameternya. Pengaturan parameter ini menyebabkan pemuat massal menggunakan semua sumber daya CPU yang tersedia saat dijalankan. Umumnya dibutuhkan 60% -70% dari kapasitas CPU untuk menjaga operasi berjalan secepat I/O kendala memungkinkan.

  > **Note**
  >
  > Ketika `parallelism` disetel ke `OVERSUBSCRIBE` atau `HIGH` (pengaturan default), ada risiko saat memuat data OpenCypher bahwa utas mungkin mengalami kondisi balapan dan kebuntuan, yang mengakibatkan kesalahan. `LOAD_DATA_DEADLOCK` Dalam hal ini, atur `parallelism` ke pengaturan yang lebih rendah dan coba lagi beban.

- Jika pekerjaan pemuatan Anda akan mencakup beberapa permintaan pemuatan, gunakan parameter `queueRequest`. Mengatur `queueRequest` ke `TRUE` memungkinkan Neptune mengantrekan permintaan Anda sehingga Anda tidak perlu menunggu sampai satu permintaan selesai sebelum mengeluarkan permintaan yang lain.
- Jika permintaan pemuatan Anda sedang mengantre, Anda dapat mengatur tingkat ketergantungan menggunakan parameter `dependencies`, sehingga kegagalan satu pekerjaan menyebabkan pekerjaan yang bergantung gagal. Hal ini dapat mencegah inkonsistensi dalam data yang dimuat.
- Jika pekerjaan pemuatan akan melibatkan proses memperbarui nilai yang sebelumnya dimuat, pastikan untuk mengatur parameter `updateSingleCardinalityProperties` ke `TRUE`. Jika tidak, loader akan memperlakukan upaya untuk memperbarui nilai kardinalitas tunggal yang ada sebagai kesalahan. Untuk data Gremlin, kardinalitas juga ditentukan dalam header kolom properti (lihat Header Kolom Properti).

  > **Note**
  >
  > `updateSingleCardinalityProperties`Parameter tidak tersedia untuk data Resource Description Framework (RDF).

- Anda dapat menggunakan parameter `failOnError` untuk menentukan apakah operasi pemuatan massal harus gagal atau lanjut ketika kesalahan ditemui. Juga, Anda dapat menggunakan `mode` untuk memastikan bahwa pekerjaan pemuatan berlanjut memuat dari titik di mana pekerjaan sebelumnya gagal alih-alih memuat ulang data yang sudah dimuat.

- Scale up — Atur instance penulis cluster DB Anda ke ukuran maksimum sebelum pemuatan massal. Perhatikan bahwa jika Anda melakukan ini, Anda harus meningkatkan instance read-replica di cluster DB juga, atau menghapusnya sampai Anda selesai memuat data.

  Ketika pemuatan massal Anda selesai, pastikan untuk menskalakan turun instance penulis kembali.

> **Important**
>
> Jika Anda mengalami siklus restart replika baca berulang karena kelambatan replikasi selama pemuatan massal, replika Anda kemungkinan tidak dapat mengikuti penulis di cluster DB Anda. Baik skala pembaca menjadi lebih besar dari penulis, atau hapus sementara selama pemuatan massal dan kemudian buat ulang setelah selesai.

Lihat Parameter Permintaan untuk rincian lebih lanjut tentang pengaturan parameter permintaan loader.

Referensi Loader Neptune

Bagian ini menjelaskan `Loader` APIs untuk Amazon Neptunus yang tersedia dari titik akhir HTTP dari instans DB Neptunus.

> **Note**
>
> Lihat Pesan Kesalahan dan Umpan Neptune Loader untuk daftar pesan kesalahan dan umpan yang dikembalikan oleh loader jika terjadi kesalahan.

Daftar Isi

- Perintah Loader Neptune
  - Sintaks Permintaan Neptune
  - Parameter Permintaan Loader Neptune
    - Pertimbangan khusus untuk memuat data OpenCypher
  - Sintaks Respons Loader Neptune
  - Kesalahan Loader Neptune

- Contoh Neptune Loader
- Neptunus Loader API Get-Status
  - Permintaan Neptune Loader Get-Status
    - Sintaks Get-Status permintaan loader
    - Parameter permintaan Neptune Loader Get-Status
  - Tanggapan Neptunus Loader Get-Status
    - Tata letak JSON Respons Pemuat Neptunus Get-Status
    - Neptunus Loader Get-Status overallStatus dan objek respons failedFeeds
    - Objek respons Get-Status errors Neptunus Loader
    - Objek respons Get-Status errorLogs Neptunus Loader
  - Contoh Pemuat Get-Status Neptunus
    - Contoh permintaan untuk status pemuatan
    - Contoh permintaan untuk loadIds
    - Contoh permintaan untuk status terperinci
  - Contoh Neptunus Loader Get-Status errorLogs
    - Contoh respons status rinci ketika terjadi kesalahan
    - Contoh kesalahan Data prefetch task interrupted
- Pembatalan Pekerjaan Neptune Loader
  - Sintaks permintaan Batalkan Pekerjaan
  - Parameter Permintaan Batalkan Pekerjaan
  - Sintaks Respons Batalkan Pekerjaan
  - Kesalahan Batalkan Pekerjaan
  - Pesan Kesalahan Batalkan Pekerjaan
  - Contoh Pembatalan Pekerjaan

Perintah Loader Neptune

Memuat data dari bucket Amazon S3 ke instans DB Neptune.

Untuk memuat data, Anda harus mengirim permintaan `POST` HTTP ke titik akhir `https://`*`your-neptune-endpoint`*`:`*`port`*`/loader`. Parameter untuk permintaan `loader` dapat dikirim dalam badan `POST` atau sebagai parameter URL yang dikodekan.

> **Important**
>
> Tipe MIME harus `application/json`.

Bucket Amazon S3 harus berada di AWS Wilayah yang sama dengan cluster.

> **Note**
>
> Anda dapat memuat data terenkripsi dari Amazon S3 jika dienkripsi menggunakan Mode `SSE-S3` Amazon S3. Dalam hal ini, Neptune dapat meniru kredensial Anda dan mengeluarkan panggilan `s3:getObject` atas nama Anda.
> Anda juga dapat memuat data terenkripsi dari Amazon S3 yang dienkripsi menggunakan `SSE-KMS`, selama IAM role Anda mencakup izin yang diperlukan untuk mengakses AWS KMS. Tanpa AWS KMS izin yang tepat, operasi beban massal gagal dan mengembalikan `LOAD_FAILED` respons.
> Neptune saat ini tidak mendukung pemuatan data yang dienkripsi Amazon S3 menggunakan Mode `SSE-C`.

Anda tidak perlu menunggu untuk satu pekerjaan pemuatan selesai sebelum Anda memulai satu pekerjaan lagi. Neptune dapat mengantrekan sebanyak 64 permintaan pekerjaan sekaligus, asalkan parameter `queueRequest` mereka semua diatur ke `"TRUE"`. Urutan antrian pekerjaan akan menjadi first-in-first-out (FIFO). Jika Anda tidak ingin pekerjaan pemuatan diantrian, di sisi lain, Anda dapat mengatur `queueRequest` parameternya ke `"FALSE"` (default), sehingga pekerjaan pemuatan akan gagal jika yang lain sudah dalam proses.

Anda dapat menggunakan parameter `dependencies` untuk mengantrekan pekerjaan yang hanya harus dijalankan setelah pekerjaan sebelumnya yang ditentukan dalam antrian telah berhasil diselesaikan. Jika Anda melakukan itu dan salah satu dari mereka pekerjaan tertentu gagal, pekerjaan Anda tidak akan dijalankan dan statusnya akan diatur ke `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

Sintaks Permintaan Neptune

```
{
  "source" : "string",
  "format" : "string",
  "iamRoleArn" : "string",
```

```
  "mode": "NEW|RESUME|AUTO",
  "region" : "us-east-1",
  "failOnError" : "string",
  "parallelism" : "string",
  "parserConfiguration" : {
    "baseUri" : "http://base-uri-string",
    "namedGraphUri" : "http://named-graph-string"
  },
  "updateSingleCardinalityProperties" : "string",
  "queueRequest" : "TRUE",
  "dependencies" : ["load_A_id", "load_B_id"]
}
```

edgeOnlyLoad Sintaks

Untuk an`edgeOnlyLoad`, sintaksnya adalah:

```
{
"source" : "string",
"format" : "string",
"iamRoleArn" : "string",
"mode": "NEW|RESUME|AUTO",
"region" : "us-east-1",
"failOnError" : "string",
"parallelism" : "string",
"edgeOnlyLoad" : "string",
"parserConfiguration" : {
    "baseUri" : "http://base-uri-string",
    "namedGraphUri" : "http://named-graph-string"
},
"updateSingleCardinalityProperties" : "string",
"queueRequest" : "TRUE",
"dependencies" : ["load_A_id", "load_B_id"]
}
```

Parameter Permintaan Loader Neptune

- **`source`** — Sebuah URI Amazon S3.

  Parameter `SOURCE` menerima URI Amazon S3 yang mengidentifikasi satu file, beberapa file, folder, atau beberapa folder. Neptune memuat setiap file data dalam folder yang ditentukan.

  URI dapat berupa format berikut.

- `s3://bucket_name/object-key-name`
- `https://s3.amazonaws.com/bucket_name/object-key-name`
- `https://s3.us-east-1.amazonaws.com/bucket_name/object-key-name`

`object-key-name`Elemen URI setara dengan parameter awalan dalam panggilan API Amazon ListObjectsS3. Ini mengidentifikasi semua objek dalam bucket Amazon S3 yang ditentukan yang namanya dimulai dengan prefiks itu. Itu bisa berupa satu file atau folder, atau beberapa and/or folder file.

Folder atau folder-folder yang ditentukan dapat berisi beberapa file vertex dan beberapa file edge.

Misalnya, jika Anda memiliki struktur folder dan file berikut di bucket Amazon S3 bernama: `bucket-name`

```
s3://bucket-name/a/bc
s3://bucket-name/ab/c
s3://bucket-name/ade
s3://bucket-name/bcd
```

Jika parameter sumber ditentukan sebagai`s3://bucket-name/a`, tiga file pertama akan dimuat.

```
s3://bucket-name/a/bc
s3://bucket-name/ab/c
s3://bucket-name/ade
```

- **`format`** — Format data. Untuk informasi selengkapnya tentang format data untuk perintah `Loader` Neptune, lihat Menggunakan pemuat massal Amazon Neptunus untuk menelan data.

  Nilai yang diizinkan

  - **`csv`**untuk format data CSV Gremlin.
  - **`opencypher`**untuk format data CSV OpenCypher.
  - **`ntriples`**untuk format data RDF N-Triples.
  - **`nquads`**untuk format data N-Quads RDF.
  - **`rdfxml`**untuk format data RDF\ XHTML RDF.
  - **`turtle`**untuk format data Turtle RDF.
- **`iamRoleArn`** — Amazon Resource Name (ARN) untuk IAM role yang akan diasumsikan oleh instans DB Neptune untuk akses ke bucket S3. Untuk informasi tentang membuat peran yang

memiliki akses ke Amazon S3 dan kemudian mengaitkannya dengan klaster Neptune, lihat Prasyarat: IAM role dan Akses Amazon S3.

Dimulai dengan rilis engine 1.2.1.0.R3, Anda juga dapat menghubungkan beberapa peran IAM jika instans DB Neptunus dan bucket Amazon S3 berada di Akun yang berbeda. AWS Dalam hal ini, `iamRoleArn` berisi daftar peran yang dipisahkan koma ARNs, seperti yang dijelaskan dalam. Merantai peran IAM di Amazon Neptunus Contoh:

```
curl -X POST https://localhost:8182/loader \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://(the target bucket name)/(the target date file name)",
        "iamRoleArn" : "arn:aws:iam::(Account A
 ID):role/(RoleA),arn:aws:iam::(Account B ID):role/(RoleB),arn:aws:iam::(Account C
 ID):role/(RoleC)",
        "format" : "csv",
        "region" : "us-east-1"
      }'
```

- **`region`**— `region` Parameter harus cocok dengan AWS Wilayah cluster dan bucket S3.

  Amazon Neptune tersedia di Wilayah berikut:

  - US East (N. Virginia): `us-east-1`
  - AS Timur (Ohio): `us-east-2`
  - US West (N. California): `us-west-1`
  - US West (Oregon): `us-west-2`
  - Canada (Central): `ca-central-1`
  - Kanada Barat (Calgary): `ca-west-1`
  - South America (São Paulo): `sa-east-1`
  - Eropa (Stockholm): `eu-north-1`
  - Eropa (Spanyol): `eu-south-2`
  - Eropa (Irlandia): `eu-west-1`
  - Eropa (London): `eu-west-2`
  - Eropa (Paris): `eu-west-3`
  - Eropa (Frankfurt): `eu-central-1`
  - Timur Tengah (Bahrain): `me-south-1`

- Timur Tengah (UEA): `me-central-1`
- Israel (Tel Aviv): `il-central-1`
- Afrika (Cape Town): `af-south-1`
- Asia Pasifik (Hong Kong): `ap-east-1`
- Asia Pacific (Tokyo): `ap-northeast-1`
- Asia Pasifik (Seoul): `ap-northeast-2`
- Asia Pasifik (Osaka): `ap-northeast-3`
- Asia Pacific (Singapore): `ap-southeast-1`
- Asia Pacific (Sydney): `ap-southeast-2`
- Asia Pasifik (Jakarta): `ap-southeast-3`
- Asia Pasifik (Melbourne): `ap-southeast-4`
- Asia Pasifik (Malaysia): `ap-southeast-5`
- Asia Pasifik (Mumbai): `ap-south-1`
- Asia Pasifik (Hyderabad): `ap-south-2`
- Tiongkok (Beijing): `cn-north-1`
- Tiongkok (Ningxia): `cn-northwest-1`
- AWS GovCloud (AS-Barat): `us-gov-west-1`
- AWS GovCloud (AS-Timur): `us-gov-east-1`

- **`mode`** — Mode pekerjaan pemuatan.

  Nilai yang diizinkan:`RESUME`,`NEW`,`AUTO`.

  Nilai default: `AUTO`

  - `RESUME` — Dalam mode RESUME, loader mencari pemuatan sebelumnya dari sumber ini, dan jika menemukan satu, melanjutkan pekerjaan pemuatan tersebut. Jika tidak ada pekerjaan pemuatan sebelumnya yang ditemukan, loader berhenti.

    Loader menghindari memuat ulang file yang berhasil dimuat di pekerjaan sebelumnya. Ia hanya mencoba untuk memproses file yang gagal. Jika Anda menjatuhkan data yang dimuat sebelumnya dari klaster Neptune Anda, data tersebut tidak dimuat ulang dalam mode ini. Jika pekerjaan pemuatan sebelumnya berhasil memuat semua file dari sumber yang sama, tidak ada yang dimuat ulang, dan loader mengembalikan keberhasilan.

- `NEW` — Dalam mode NEW, it menciptakan permintaan pemuatan baru terlepas dari pemuatan sebelumnya. Anda dapat menggunakan mode ini untuk memuat ulang semua data dari sumber setelah menjatuhkan data yang dimuat sebelumnya dari klaster Neptune Anda, atau untuk memuat data baru yang tersedia di sumber yang sama.
- `AUTO` — Dalam mode AUTO, loader mencari pekerjaan pemuatan sebelumnya dari sumber yang sama, dan jika menemukannya, melanjutkan pekerjaan itu, seperti pada mode `RESUME`.

  Jika loader tidak menemukan pekerjaan pemuatan sebelumnya dari sumber yang sama, loader akan memuat semua data dari sumbernya, seperti pada mode `NEW`.

- **`edgeOnlyLoad`**- Bendera yang mengontrol urutan pemrosesan file selama pemuatan massal.

  Nilai yang diizinkan: `"TRUE"`, `"FALSE"`.

  Nilai default:`"FALSE"`.

  Ketika parameter ini diatur ke “FALSE”, loader secara otomatis memuat file simpul terlebih dahulu, lalu file tepi sesudahnya. Ini dilakukan dengan terlebih dahulu memindai semua file untuk menentukan isinya (simpul atau tepi). Ketika parameter ini diatur ke “TRUE”, loader melewatkan fase pemindaian awal dan segera memuat semua file sesuai urutan kemunculannya. Untuk informasi selengkapnya, lihat optimasi beban massal.

- **`failOnError`** — Sebuah bendera untuk mengubah berhenti penuh pada kesalahan.

  Nilai yang diizinkan: `"TRUE"`, `"FALSE"`.

  Nilai default:`"TRUE"`.

  Ketika parameter ini diatur ke `"FALSE"`, loader mencoba memuat semua data di lokasi yang ditentukan, melewatkan entri apa pun yang memiliki kesalahan.

  Ketika parameter ini diatur ke `"TRUE"`, loader berhenti segera setelah menemukan kesalahan. Data yang dimuat sampai saat itu tetap ada.

- **`parallelism`** — Ini adalah parameter opsional yang dapat diatur untuk mengurangi jumlah utas yang digunakan oleh proses pemuatan massal.

  Nilai yang diizinkan:

  - `LOW`— Jumlah benang yang digunakan adalah jumlah v yang tersedia CPUs dibagi 8.
  - `MEDIUM`— Jumlah benang yang digunakan adalah jumlah v yang tersedia CPUs dibagi 2.
  - `HIGH`— Jumlah thread yang digunakan sama dengan jumlah yang tersedia vCPUs.

- `OVERSUBSCRIBE`— Jumlah utas yang digunakan adalah jumlah v yang tersedia CPUs dikalikan dengan 2. Jika nilai ini digunakan, loader massal mengambil semua sumber daya yang tersedia.

  Ini tidak berarti, bagaimanapun, bahwa pengaturan `OVERSUBSCRIBE` menghasilkan 100% utilisasi CPU. Karena operasi beban I/O terikat, pemanfaatan CPU tertinggi yang diharapkan adalah dalam kisaran 60% hingga 70%.

Nilai default: `HIGH`

`parallelism`Pengaturan terkadang dapat mengakibatkan kebuntuan antar utas saat memuat data OpenCypher. Ketika ini terjadi, Neptunus mengembalikan kesalahan. `LOAD_DATA_DEADLOCK` Anda biasanya dapat memperbaiki masalah dengan mengatur `parallelism` ke pengaturan yang lebih rendah dan mencoba kembali perintah load.

- **`parserConfiguration`** — Sebuah objek opsional dengan nilai konfigurasi parser tambahan. Masing-masing parameter turunan juga opsional:

Nama	Nilai Contoh	Deskripsi
`namedGraphUri`	*`http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`*	Grafik default untuk semua format RDF ketika tidak ada grafik yang ditentukan (untuk format non-quads dan entri NQUAD tanpa grafik). Bawaannya adalah `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`
`baseUri`	*`http://aws.amazon.com/neptune/default`*	URI dasar untuk RDF/XML dan format Turtle. Nilai default-nya `http://aws.amazon.com/neptune/default` .
`allowEmptyStrings`	*`true`*	Pengguna Gremlin harus dapat melewatkan nilai string kosong (“”) sebagai properti node dan edge saat memuat

data CSV. Jika `allowEmptyStrings` diatur ke `false` (default), string kosong diperlakukan sebagai null dan tidak dimuat.

Jika `allowEmptyStrings` diatur ke `true`, loader memperlakukan string kosong sebagai nilai properti yang valid dan memuatnya sesuai keperluan.

Untuk informasi selengkapnya, lihat Grafik Standar SPARQL dan Grafik Bernama.

- **`updateSingleCardinalityProperties`** — Ini adalah parameter opsional yang mengontrol bagaimana bulk loader memperlakukan nilai baru untuk properti vertex atau edge single-cardinality. Ini tidak didukung untuk memuat data OpenCypher (lihat). Memuat data OpenCypher

  Nilai yang diizinkan: `"TRUE"`, `"FALSE"`.

  Nilai default:`"FALSE"`.

  Secara default, atau saat `updateSingleCardinalityProperties` secara eksplisit diatur ke `"FALSE"`, loader memperlakukan nilai baru sebagai kesalahan, karena melanggar kardinalitas tunggal.

  Saat `updateSingleCardinalityProperties` diatur ke `"TRUE"`, di sisi lain, loader massal menggantikan nilai yang ada dengan yang baru. Jika beberapa edge atau nilai properti vertex single-cardinality disediakan dalam file sumber yang dimuat, nilai akhir pada akhir pemuatan massal bisa menjadi salah satu dari nilai-nilai baru tersebut. Loader hanya menjamin bahwa nilai yang ada telah digantikan oleh salah satu yang baru.
- **`queueRequest`** — Ini adalah parameter bendera opsional yang menunjukkan apakah permintaan pemuatan dapat diantrekan atau tidak.

  Anda tidak perlu menunggu satu pekerjaan muat selesai sebelum mengeluarkan pekerjaan berikutnya, karena Neptune dapat mengantrekan sebanyak 64 pekerjaan sekaligus, asalkan

parameter `queueRequest` semua diatur ke `"TRUE"`. Urutan antrian pekerjaan akan menjadi first-in-first-out (FIFO).

Jika parameter `queueRequest` dihilangkan atau diatur ke `"FALSE"`, permintaan pemuatan akan gagal jika pekerjaan pemuatan lain sudah berjalan.

Nilai yang diizinkan: `"TRUE"`, `"FALSE"`.

Nilai default:`"FALSE"`.

- **`dependencies`** — Ini adalah parameter opsional yang dapat membuat permintaan pemuatan yang mengantre bergantung pada penyelesaian yang berhasil dari satu atau lebih pekerjaan sebelumnya dalam antrean.

  Neptune dapat mengantrekan sebanyak 64 permintaan pemuatan sekaligus, jika parameter `queueRequest` permintaannya diatur ke `"TRUE"`. Parameter `dependencies` memungkinkan Anda melakukan eksekusi seperti permintaan mengantre yang tergantung pada penyelesaian yang berhasil dari satu atau lebih permintaan ditentukan sebelumnya dalam antrean.

  Misalnya, jika pemuatan `Job-A` dan `Job-B` independen satu sama lain, namun pemuatan `Job-C` membutuhkan `Job-A` dan `Job-B` harus selesai sebelum dimulai, lanjutkan sebagai berikut:

  1. Kirim `load-job-A` dan `load-job-B` satu demi satu dalam urutan apa pun, dan simpan load-id mereka.
  2. Kirim `load-job-C` dengan load-id dari dua pekerjaan di bidang `dependencies`-nya:

  ```
  "dependencies" : ["job_A_load_id", "job_B_load_id"]
  ```

  Karena parameter `dependencies`, loader massal tidak akan memulai `Job-C` sampai `Job-A` dan `Job-B` telah berhasil diselesaikan. Jika salah satu dari mereka gagal, Job-C tidak akan dieksekusi, dan statusnya akan diatur ke `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

  Anda dapat mengatur beberapa tingkat ketergantungan dengan cara ini, sehingga kegagalan satu pekerjaan akan menyebabkan semua permintaan yang secara langsung atau tidak langsung tergantung padanya untuk dibatalkan.

- **`userProvidedEdgeIds`**- Parameter ini diperlukan hanya ketika memuat data OpenCypher yang berisi hubungan. IDs Itu harus disertakan dan diatur ke `True` ketika hubungan OpenCypher secara eksplisit IDs disediakan dalam data beban (disarankan).

Ketika tidak `userProvidedEdgeIds` ada atau diatur ke`True`, `:ID` kolom harus ada di setiap file hubungan dalam beban.

Ketika `userProvidedEdgeIds` hadir dan diatur ke`False`, file hubungan dalam beban tidak boleh berisi `:ID` kolom. Sebagai gantinya, pemuat Neptunus secara otomatis menghasilkan ID untuk setiap hubungan.

Sangat berguna untuk memberikan hubungan secara IDs eksplisit sehingga loader dapat melanjutkan pemuatan setelah kesalahan dalam data CSV telah diperbaiki, tanpa harus memuat ulang hubungan apa pun yang telah dimuat. Jika hubungan IDs belum ditetapkan secara eksplisit, loader tidak dapat melanjutkan pemuatan yang gagal jika ada file hubungan yang harus diperbaiki, dan sebagai gantinya harus memuat ulang semua hubungan.

- `accessKey` — [usang] Access key ID dari IAM role dengan akses ke bucket S3 dan file data.

  Parameter `iamRoleArn` dianjurkan sebagai gantinya. Untuk informasi tentang membuat peran yang memiliki akses ke Amazon S3 dan kemudian mengaitkannya dengan klaster Neptune, lihat Prasyarat: IAM role dan Akses Amazon S3.

  Untuk informasi lebih lanjut, lihat Access key (access key ID dan secret access key).

- `secretKey` – [usang] Parameter `iamRoleArn` dianjurkan sebagai gantinya. Untuk informasi tentang membuat peran yang memiliki akses ke Amazon S3 dan kemudian mengaitkannya dengan klaster Neptune, lihat Prasyarat: IAM role dan Akses Amazon S3.

  Untuk informasi lebih lanjut, lihat Access key (access key ID dan secret access key).

Pertimbangan khusus untuk memuat data OpenCypher

- Saat memuat data OpenCypher dalam format CSV, parameter format harus diatur ke. `opencypher`
- `updateSingleCardinalityProperties`Parameter tidak didukung untuk beban OpenCypher karena semua properti OpenCypher memiliki kardinalitas tunggal. Format beban OpenCypher tidak mendukung array, dan jika nilai ID muncul lebih dari sekali, itu diperlakukan sebagai duplikat atau kesalahan penyisipan (lihat di bawah).
- Pemuat Neptunus menangani duplikat yang ditemuinya dalam data OpenCypher sebagai berikut:
  - Jika loader menemukan beberapa baris dengan ID node yang sama, mereka digabungkan menggunakan aturan berikut:

    - Semua label di baris ditambahkan ke node.
    - Untuk setiap properti, hanya satu dari nilai properti yang dimuat. Pemilihan yang akan dimuat adalah non-deterministik.
  - Jika loader menemukan beberapa baris dengan ID hubungan yang sama, hanya satu dari mereka yang dimuat. Pemilihan yang akan dimuat adalah non-deterministric.
  - Loader tidak pernah memperbarui nilai properti dari node atau relasi yang ada dalam database jika menemukan data beban yang memiliki ID dari node atau relasi yang ada. Namun, itu memuat label node dan properti yang tidak ada di node atau hubungan yang ada.
- Meskipun Anda tidak harus menetapkan IDs hubungan, biasanya itu ide yang bagus (lihat `userProvidedEdgeIds` parameter di atas). Tanpa hubungan eksplisit IDs, pemuat harus memuat ulang semua hubungan jika terjadi kesalahan dalam file hubungan, daripada melanjutkan pemuatan dari tempat gagal.

  Selain itu, jika data pemuatan tidak berisi hubungan eksplisit IDs, loader tidak memiliki cara untuk mendeteksi hubungan duplikat.

Berikut adalah contoh perintah beban OpenCypher:

```
curl -X POST https://your-neptune-endpoint:port/loader \
     -H 'Content-Type: application/json' \
     -d '
     {
       "source" : "s3://bucket-name/object-key-name",
       "format" : "opencypher",
       "userProvidedEdgeIds": "TRUE",
       "iamRoleArn" : "arn:aws:iam::account-id:role/role-name",
       "region" : "region",
       "failOnError" : "FALSE",
       "parallelism" : "MEDIUM",
     }'
```

Respons loader sama dengan normal. Contoh:

```
{
  "status" : "200 OK",
  "payload" : {
    "loadId" : "guid_as_string"
  }
```

```
}
```

Sintaks Respons Loader Neptune

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "guid_as_string"
    }
}
```

200 OK

Pekerjaan pemuatan yang berhasil dimulai mengembalikan kode `200`.

Kesalahan Loader Neptune

Ketika terjadi kesalahan, objek JSON dikembalikan dalam `BODY` respons. Objek `message` berisi deskripsi kesalahan.

Kategori Kesalahan

- `Error 400` - kesalahan Sintaks mengembalikan kesalahan permintaan buruk HTTP `400`. Pesan ini menjelaskan kesalahan.
- `Error 500` — Permintaan valid yang tidak dapat diproses mengembalikan HTTP `500` kesalahan server internal. Pesan ini menjelaskan kesalahan.

Berikut ini adalah kemungkinan pesan kesalahan dari loader dengan deskripsi kesalahannya.

Pesan Kesalahan Loader

- `Couldn't find the AWS credential for iam_role_arn` (HTTP 400)

  Kredensialnya tidak ditemukan. Verifikasi kredensyal yang disediakan terhadap konsol atau output IAM. AWS CLI Pastikan Anda telah menambahkan peran IAM yang ditentukan `iamRoleArn` ke dalam cluster.
- `S3 bucket not found for source` (HTTP 400)

  Bucket S3 tidak ada. Periksa nama bucket.
- `The source source-uri does not exist/not reachable` (HTTP 400)

File yang cocok tidak ditemukan di bucket S3.

- `Unable to connect to S3 endpoint. Provided source = source-uri and region = aws-region` (HTTP 500)

  Tidak dapat terhubung ke Amazon S3. Wilayah harus sesuai dengan Wilayah klaster. Pastikan bahwa Anda memiliki VPC Endpoint. Untuk informasi tentang cara membuat VPC Endpoint, lihat Membuat VPC Endpoint Amazon S3.

- `Bucket is not in provided Region (aws-region)` (HTTP 400)

  Bucket harus berada di AWS Wilayah yang sama dengan instans DB Neptunus Anda.

- `Unable to perform S3 list operation` (HTTP 400)

  Pengguna atau IAM role tidak memiliki izin `List` pada bucket atau folder. Periksa kebijakan atau daftar kontrol akses (ACL) di bucket.

- `Start new load operation not permitted on a read replica instance` (HTTP 405)

  Memuat adalah operasi menulis. Coba lagi muat pada titik akhir read/write cluster.

- `Failed to start load because of unknown error from S3` (HTTP 500)

  Amazon S3 mengembalikan kesalahan yang tidak diketahui. Hubungi AWS Dukungan.

- `Invalid S3 access key` (HTTP 400)

  Access key tidak valid. Periksa kredensial yang disediakan.

- `Invalid S3 secret key` (HTTP 400)

  Kunci rahasia tidak valid. Periksa kredensial yang disediakan.

- `Max concurrent load limit breached` (HTTP 400)

  Jika permintaan pemuatan diajukan tanpa `"queueRequest" : "TRUE"`, dan pekerjaan pemuatan saat ini berjalan, permintaan akan gagal dengan kesalahan ini.

- `Failed to start new load for the source "source name". Max load task queue size limit breached. Limit is 64` (HTTP 400)

  Neptune mendukung antrean sebanyak 64 pekerjaan loader sekaligus. Jika permintaan pemuatan tambahan diserahkan ke antrean ketika sudah berisi 64 pekerjaan, permintaan gagal dengan pesan ini.

Contoh Neptune Loader

Contoh ini menunjukkan cara menggunakan loader Neptunus untuk memuat data ke dalam database grafik Neptunus menggunakan format CSV Gremlin. Permintaan dikirim sebagai permintaan HTTP POST ke titik akhir loader Neptunus, dan badan permintaan berisi parameter yang diperlukan untuk menentukan sumber data, format, peran IAM, dan opsi konfigurasi lainnya. Respons termasuk ID beban, yang dapat digunakan untuk melacak kemajuan proses pemuatan data.

Example Permintaan

Berikut ini adalah permintaan yang dikirim melalui HTTP POST menggunakan perintah `curl`. Permintaan tersebut memuatkan file dalam format CSV Neptune. Untuk informasi selengkapnya, lihat Format data muat Gremlin.

```
curl -X POST \
    -H 'Content-Type: application/json' \
    https://your-neptune-endpoint:port/loader -d '
    {
      "source" : "s3://bucket-name/object-key-name",
      "format" : "csv",
      "iamRoleArn" : "ARN for the IAM role you are using",
      "region" : "region",
      "failOnError" : "FALSE",
      "parallelism" : "MEDIUM",
      "updateSingleCardinalityProperties" : "FALSE",
      "queueRequest" : "FALSE"
    }'
```

Example Respons

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

Neptunus Loader API Get-Status

Mendapat status pekerjaan `loader`.

Untuk mendapatkan status pemuatan, Anda harus mengirim permintaan GET HTTP ke titik akhir `https://your-neptune-endpoint:port/loader`. Untuk mendapatkan status permintaan pemuatan tertentu, Anda harus menyertakan `loadId` sebagai parameter URL, atau menambahkan `loadId` ke jalur URL.

Neptune hanya melacak 1.024 pekerjaan pemuatan massal terbaru, dan hanya menyimpan 10.000 rincian kesalahan terakhir per pekerjaan.

Lihat Pesan Kesalahan dan Umpan Neptune Loader untuk daftar pesan kesalahan dan umpan yang dikembalikan oleh loader jika terjadi kesalahan.

Daftar Isi

- Permintaan Neptune Loader Get-Status
  - Sintaks Get-Status permintaan loader
  - Parameter permintaan Neptune Loader Get-Status
- Tanggapan Neptunus Loader Get-Status
  - Tata letak JSON Respons Pemuat Neptunus Get-Status
  - Neptunus Loader Get-Status overallStatus dan objek respons failedFeeds
  - Objek respons Get-Status errors Neptunus Loader
  - Objek respons Get-Status errorLogs Neptunus Loader
- Contoh Pemuat Get-Status Neptunus
  - Contoh permintaan untuk status pemuatan
  - Contoh permintaan untuk loadIds
  - Contoh permintaan untuk status terperinci
- Contoh Neptunus Loader Get-Status errorLogs
  - Contoh respons status rinci ketika terjadi kesalahan
  - Contoh kesalahan Data prefetch task interrupted

Permintaan Neptune Loader Get-Status

Sintaks Get-Status permintaan loader

```
GET https://your-neptune-endpoint:port/loader?loadId=loadId
```

```
GET https://your-neptune-endpoint:port/loader/loadId
```

```
GET https://your-neptune-endpoint:port/loader
```

Parameter permintaan Neptune Loader Get-Status

- **`loadId`**— ID tugas pemuatan. Jika Anda tidak menentukan`loadId`, daftar beban IDs dikembalikan.
- **`details`**— Sertakan detail di luar status keseluruhan.

  Nilai yang diizinkan: `TRUE`, `FALSE`.

  Nilai default:`FALSE`.
- **`errors`**— Sertakan daftar kesalahan.

  Nilai yang diizinkan: `TRUE`, `FALSE`.

  Nilai default:`FALSE`.

  Daftar kesalahan dipecah dalam beberapa bagian. Parameter `page` dan `errorsPerPage` memungkinkan Anda untuk melalui semua kesalahan dalam beberapa bagian.
- **`page`**— Nomor halaman kesalahan. Hanya berlaku dengan parameter `errors` diatur ke `TRUE`.

  Nilai yang diizinkan: Bilangan bulat positif.

  Nilai default: 1.
- **`errorsPerPage`**— Jumlah kesalahan per setiap halaman. Hanya berlaku dengan parameter `errors` diatur ke `TRUE`.

  Nilai yang diizinkan: Bilangan bulat positif.

  Nilai default: 10.
- **`limit`**— Jumlah id pemuatan untuk dicantumkan. Hanya berlaku saat meminta daftar pemuatan IDs dengan mengirimkan `GET` permintaan tanpa `loadId` ditentukan.

  Nilai yang diizinkan: Bilangan bulat positif dari 1 hingga 100.

  Nilai default: 100.

- **`includeQueuedLoads`**- Parameter opsional yang dapat digunakan untuk mengecualikan beban IDs permintaan beban antrian ketika daftar beban IDs diminta.

  Secara default, beban IDs semua pekerjaan pemuatan dengan status `LOAD_IN_QUEUE` termasuk dalam daftar seperti itu. Mereka muncul sebelum beban IDs pekerjaan lain, diurutkan berdasarkan waktu mereka ditambahkan ke antrian dari yang terbaru hingga yang paling awal.

  Nilai yang diizinkan: `TRUE`, `FALSE`.

  Nilai default:`TRUE`.

Tanggapan Neptunus Loader Get-Status

Contoh respons berikut dari Get-Status Neptunus API menjelaskan keseluruhan struktur respons, menjelaskan berbagai bidang dan tipe datanya, serta penanganan kesalahan dan detail log kesalahan.

Tata letak JSON Respons Pemuat Neptunus Get-Status

Tata letak umum respon status loader adalah sebagai berikut:

```
{
    "status" : "200 OK",
    "payload" : {
        "feedCount" : [
            {
                "LOAD_FAILED" : number
            }
        ],
        "overallStatus" : {
            "fullUri" : "s3://bucket/key",
            "runNumber" : number,
            "retryNumber" : number,
            "status" : "string",
            "totalTimeSpent" : number,
            "startTime" : number,
            "totalRecords" : number,
            "totalDuplicates" : number,
            "parsingErrors" : number,
            "datatypeMismatchErrors" : number,
            "insertErrors" : number,
        },
```

```
        "failedFeeds" : [
            {
                "fullUri" : "s3://bucket/key",
                "runNumber" : number,
                "retryNumber" : number,
                "status" : "string",
                "totalTimeSpent" : number,
                "startTime" : number,
                "totalRecords" : number,
                "totalDuplicates" : number,
                "parsingErrors" : number,
                "datatypeMismatchErrors" : number,
                "insertErrors" : number,
            }
        ],
        "errors" : {
            "startIndex" : number,
            "endIndex" : number,
            "loadId" : "string,
            "errorLogs" : [ ]
        }
    }
}
```

Neptunus Loader Get-Status **`overallStatus`** dan objek respons **`failedFeeds`**

Respons yang mungkin dikembalikan untuk setiap umpan gagal, termasuk deskripsi kesalahan, adalah sama seperti untuk objek `overallStatus` dalam respons `Get-Status`.

Bidang berikut ini muncul di objek `overallStatus` untuk semua pemuatan, dan objek `failedFeeds` untuk setiap umpan gagal:

- **`fullUri`** — URI dari file atau file-file yang akan dimuat.

  Jenis: string

  Format:`s3://`*`bucket`*`/`*`key`*.
- **`runNumber`**— Jumlah run dari pemuatan atau umpan ini. Ini bertambah ketika pemuatan dimulai ulang.

  Jenis: panjang yang tidak diberi tanda.

- **`retryNumber`**— Jumlah retry dari pemuatan atau umpan ini. Ini bertambah ketika loader secara otomatis mencoba ulang umpan atau pemuatan.

  Jenis: panjang yang tidak diberi tanda.

- **`status`**— Status yang dikembalikan dari pemuatan atau umpan. `LOAD_COMPLETED` menunjukkan pemuatan yang berhasil tanpa masalah. Untuk daftar pesan status beban lainnya, lihat. Pesan Kesalahan dan Umpan Neptune Loader

  Jenis: string.

- **`totalTimeSpent`**— Waktu, dalam hitungan detik, yang dihabiskan untuk mengurai dan memasukkan data untuk pemuatan atau umpan. Ini tidak termasuk waktu yang dihabiskan untuk mengambil daftar file sumber.

  Jenis: panjang yang tidak diberi tanda.

- **`totalRecords`**— Jumlah catatan yang dimuat atau dicoba untuk dimuat.

  Jenis: panjang yang tidak diberi tanda.

  Perhatikan bahwa saat memuat dari file CSV, jumlah catatan tidak mengacu pada jumlah baris yang dimuat, melainkan jumlah catatan individual di baris tersebut. Misalnya, ambil file CSV kecil seperti ini:

  ```
  ~id,~label,name,team
  'P-1','Player','Stokes','England'
  ```

  Neptunus akan menganggap file ini berisi 3 catatan, yaitu:

  ```
  P-1  label Player
  P-1  name  Stokes
  P-1  team  England
  ```

- **`totalDuplicates`**— Jumlah catatan duplikat yang ditemui.

  Jenis: panjang yang tidak diberi tanda.

  Seperti dalam kasus `totalRecords` penghitungan, nilai ini berisi jumlah catatan duplikat individu dalam file CSV, bukan jumlah baris duplikat. Ambil file CSV kecil ini, misalnya:

  ```
  ~id,~label,name,team
  ```

```
P-2,Player,Kohli,India
P-2,Player,Kohli,India
```

Status yang dikembalikan setelah memuatnya akan terlihat seperti ini, melaporkan 6 total catatan, 3 di antaranya adalah duplikat:

```
{
  "status": "200 OK",
  "payload": {
    "feedCount": [
      {
        "LOAD_COMPLETED": 1
      }
    ],
    "overallStatus": {
      "fullUri": "(the URI of the CSV file)",
      "runNumber": 1,
      "retryNumber": 0,
      "status": "LOAD_COMPLETED",
      "totalTimeSpent": 3,
      "startTime": 1662131463,
      "totalRecords": 6,
      "totalDuplicates": 3,
      "parsingErrors": 0,
      "datatypeMismatchErrors": 0,
      "insertErrors": 0
    }
  }
}
```

Untuk pemuatan OpenCypher, duplikat dihitung ketika:

- Loader mendeteksi bahwa baris dalam file node memiliki ID tanpa ruang ID yang sama dengan nilai ID lain tanpa ruang ID, baik di baris lain atau milik node yang ada.
- Loader mendeteksi bahwa baris dalam file node memiliki ID dengan ruang ID yang sama dengan nilai ID lain dengan ruang ID, baik di baris lain atau milik node yang ada.

Lihat Pertimbangan khusus untuk memuat data OpenCypher.

- **`parsingErrors`**— Jumlah kesalahan penguraian yang ditemui.

  Jenis: panjang yang tidak diberi tanda.

- **`datatypeMismatchErrors`**— Jumlah catatan dengan tipe data yang tidak cocok dengan data yang diberikan.

  Jenis: panjang yang tidak diberi tanda.

- **`insertErrors`**— Jumlah catatan yang tidak dapat dimasukkan karena kesalahan.

  Jenis: panjang yang tidak diberi tanda.

Objek respons Get-Status **`errors`** Neptunus Loader

Kesalahan termasuk ke dalam kategori berikut:

- **`Error 400`** – `loadId` yang tidak valid mengembalikan kesalahan permintaan buruk `400` HTTP. Pesan ini menjelaskan kesalahan.
- **`Error 500`** — Permintaan valid yang tidak dapat diproses mengembalikan HTTP `500` kesalahan server internal. Pesan ini menjelaskan kesalahan.

Lihat Pesan Kesalahan dan Umpan Neptune Loader untuk daftar pesan kesalahan dan umpan yang dikembalikan oleh loader jika terjadi kesalahan.

Ketika terjadi kesalahan, sebuah objek `errors` JSON dikembalikan dalam `BODY` respons, dengan kolom berikut:

- **`startIndex`** — Indeks kesalahan pertama yang disertakan.

  Jenis: panjang yang tidak diberi tanda.

- **`endIndex`** — Indeks kesalahan terakhir yang disertakan.

  Jenis: panjang yang tidak diberi tanda.

- **`loadId`** — ID pemuatan. Anda dapat menggunakan ID ini untuk mencetak kesalahan untuk pemuatan dengan menetapkan parameter `errors` ke `TRUE`.

  Jenis: string.

- **`errorLogs`** — Daftar kesalahan.

  Jenis: daftar.

Objek respons Get-Status **`errorLogs`** Neptunus Loader

Objek `errorLogs` di bawah `errors` dalam respons Get-Status loader berisi objek yang menggambarkan setiap kesalahan menggunakan bidang berikut:

- **`errorCode`** — Mengidentifikasi sifat kesalahan.

  Status ini dapat berupa salah satu dari nilai berikut:

  - `PARSING_ERROR`
  - `S3_ACCESS_DENIED_ERROR`
  - `FROM_OR_TO_VERTEX_ARE_MISSING`
  - `ID_ASSIGNED_TO_MULTIPLE_EDGES`
  - `SINGLE_CARDINALITY_VIOLATION`
  - `FILE_MODIFICATION_OR_DELETION_ERROR`
  - `OUT_OF_MEMORY_ERROR`
  - `INTERNAL_ERROR` (dikembalikan ketika loader massal tidak dapat menentukan jenis kesalahan).
- **`errorMessage`** — Sebuah pesan yang menjelaskan kesalahan.

  Ini bisa berupa pesan umum yang terkait dengan kode kesalahan atau pesan tertentu yang berisi detail, misalnya tentang from/to simpul yang hilang atau tentang kesalahan penguraian.
- **`fileName`** — Nama umpan.
- **`recordNum`** — Dalam kasus kesalahan penguraian, ini adalah nomor catatan dalam file catatan yang tidak dapat diurai. Hal ini diatur ke nol jika nomor catatan tidak berlaku untuk kesalahan, atau jika tidak dapat ditentukan.

Sebagai contoh, loader massal akan menghasilkan kesalahan penguraian jika mengalami baris rusak seperti berikut dalam file `nquads` RDF:

```
<http://base#subject> |http://base#predicate> <http://base#true> .
```

Seperti yang Anda lihat, `http` yang kedua pada baris di atas harus didahului oleh < ketimbang `|`. Objek kesalahan yang dihasilkan di bawah `errorLogs` dalam respon status akan terlihat seperti ini:

```
{
    "errorCode" : "PARSING_ERROR",
    "errorMessage" : "Expected '<', found: |",
```

```
    "fileName" : "s3://bucket/key",
    "recordNum" : 12345
},
```

Contoh Pemuat Get-Status Neptunus

Contoh berikut menampilkan penggunaan API Get-status loader Neptunus, yang memungkinkan Anda mengambil informasi tentang status pemuatan data Anda ke dalam database grafik Amazon Neptunus. Contoh-contoh ini mencakup tiga skenario utama: mengambil status beban tertentu, mencantumkan beban yang tersedia IDs, dan meminta informasi status terperinci untuk beban tertentu.

Contoh permintaan untuk status pemuatan

Berikut ini adalah permintaan yang dikirim melalui HTTP `GET` menggunakan perintah `curl`.

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)'
```

Example Respons

```
{
    "status" : "200 OK",
    "payload" : {
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        }
    }
}
```

Contoh permintaan untuk loadIds

Berikut ini adalah permintaan yang dikirim melalui HTTP GET menggunakan perintah `curl`.

```
curl -X GET 'https://your-neptune-endpoint:port/loader?limit=3'
```

Example Respons

```
{
    "status" : "200 OK",
    "payload" : {
         "loadIds" : [
            "a2c0ce44-a44b-4517-8cd4-1dc144a8e5b5",
            "09683a01-6f37-4774-bb1b-5620d87f1931",
            "58085eb8-ceb4-4029-a3dc-3840969826b9"
        ]
    }
}
```

Contoh permintaan untuk status terperinci

Berikut ini adalah permintaan yang dikirim melalui HTTP GET menggunakan perintah `curl`.

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)?details=true'
```

Example Respons

```
{
    "status" : "200 OK",
    "payload" : {
        "failedFeeds" : [
            {
                "datatypeMismatchErrors" : 0,
                "fullUri" : "s3://bucket/key",
                "insertErrors" : 0,
                "parsingErrors" : 5,
                "retryNumber" : 0,
                "runNumber" : 1,
                "status" : "LOAD_FAILED",
                "totalDuplicates" : 0,
                "totalRecords" : 5,
                "totalTimeSpent" : 3.0
```

```
            }
        ],
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        }
    }
}
```

Contoh Neptunus Loader Get-Status **`errorLogs`**

Contoh berikut menampilkan respons status terperinci dari pemuat Neptunus saat terjadi kesalahan selama proses pemuatan data. Contoh menggambarkan struktur respons, termasuk informasi tentang feed yang gagal, status keseluruhan, dan log kesalahan terperinci.

Contoh respons status rinci ketika terjadi kesalahan

Ini adalah permintaan yang dikirim melalui HTTP `GET` menggunakan `curl`:

```
curl -X GET 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-c29ce5f54802?details=true&errors=true&page=1&errorsPerPage=3'
```

Example dari respons terperinci ketika terjadi kesalahan

Ini adalah contoh dari respon yang mungkin Anda dapatkan dari kueri di atas, dengan objek `errorLogs` yang mencantumkan kesalahan pemuatan yang ditemui:

```
{
    "status" : "200 OK",
    "payload" : {
```

```
        "failedFeeds" : [
            {
                "datatypeMismatchErrors" : 0,
                "fullUri" : "s3://bucket/key",
                "insertErrors" : 0,
                "parsingErrors" : 5,
                "retryNumber" : 0,
                "runNumber" : 1,
                "status" : "LOAD_FAILED",
                "totalDuplicates" : 0,
                "totalRecords" : 5,
                "totalTimeSpent" : 3.0
            }
        ],
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        },
        "errors" : {
            "endIndex" : 3,
            "errorLogs" : [
                {
                    "errorCode" : "PARSING_ERROR",
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 1
                },
                {
                    "errorCode" : "PARSING_ERROR",
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
```

```
                    "recordNum" : 2
                },
                {
                    "errorCode" : "PARSING_ERROR",
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 3
                }
            ],
            "loadId" : "0a237328-afd5-4574-a0bc-c29ce5f54802",
            "startIndex" : 1
        }
    }
}
```

Contoh kesalahan **`Data prefetch task interrupted`**

Kadang-kadang ketika Anda mendapatkan status `LOAD_FAILED` dan kemudian meminta informasi lebih rinci, kesalahan yang dikembalikan mungkin berupa `PARSING_ERROR` dengan pesan `Data prefetch task interrupted`, seperti ini:

```
"errorLogs" : [
    {
        "errorCode" : "PARSING_ERROR",
        "errorMessage" : "Data prefetch task interrupted: Data prefetch task for 11467
 failed",
        "fileName" : "s3://amzn-s3-demo-bucket/some-source-file",
        "recordNum" : 0
    }
]
```

Kesalahan ini terjadi ketika ada gangguan sementara dalam proses pemuatan data yang biasanya tidak disebabkan oleh permintaan atau data Anda. Biasanya dapat diselesaikan hanya dengan menjalankan permintaan upload massal lagi. Jika Anda menggunakan pengaturan default, yaitu `"mode":"AUTO"`, dan `"failOnError":"TRUE"`, loader melewatkan file yang sudah berhasil dimuat dan melanjutkan pemuatan file yang belum dimuat saat terjadi gangguan.

Pembatalan Pekerjaan Neptune Loader

Membatalkan pekerjaan pemuatan.

Untuk memuat data, Anda harus mengirim permintaan DELETE HTTP ke titik akhir `https://your-neptune-endpoint:port/loader`. `loadId` dapat ditambahkan ke jalur URL `/loader`, atau disertakan sebagai variabel dalam URL.

Sintaks permintaan Batalkan Pekerjaan

```
DELETE https://your-neptune-endpoint:port/loader?loadId=loadId
```

```
DELETE https://your-neptune-endpoint:port/loader/loadId
```

Parameter Permintaan Batalkan Pekerjaan

loadId

ID tugas pemuatan.

Sintaks Respons Batalkan Pekerjaan

```
no response body
```

200 OK

Pekerjaan pemuatan yang berhasil dihapus mengembalikan kode `200`.

Kesalahan Batalkan Pekerjaan

Ketika terjadi kesalahan, objek JSON dikembalikan dalam `BODY` respons. Objek `message` berisi deskripsi kesalahan.

Kategori Kesalahan

- **`Error 400`** – `loadId` yang tidak valid mengembalikan kesalahan permintaan buruk `400` HTTP. Pesan ini menjelaskan kesalahan.
- **`Error 500`** — Permintaan valid yang tidak dapat diproses mengembalikan HTTP `500` kesalahan server internal. Pesan ini menjelaskan kesalahan.

Pesan Kesalahan Batalkan Pekerjaan

Berikut ini adalah kemungkinan pesan kesalahan dari API pembatalan dengan deskripsi kesalahannya.

- `The load with id = load_id does not exist or not active` (HTTP 404) — Pemuatan tidak ditemukan. Periksa nilai dari Parameter `id`.
- `Load cancellation is not permitted on a read replica instance.` (HTTP 405) - Pemuatan adalah operasi menulis. Coba lagi muat pada titik akhir read/write cluster.

Contoh Pembatalan Pekerjaan

Example Permintaan

Berikut ini adalah permintaan yang dikirim melalui HTTP `DELETE` menggunakan perintah `curl`.

```
curl -X DELETE 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-c29ce5f54802'
```

Menggunakan AWS Database Migration Service untuk memuat data ke Amazon Neptunus dari penyimpanan data yang berbeda

AWS Database Migration Service (AWS DMS) dapat memuat data ke Neptunus dari basis data sumber yang didukung dengan cepat dan aman. Database sumber tetap beroperasi penuh selama migrasi, meminimalkan waktu henti untuk aplikasi yang mengandalkannya.

Anda dapat menemukan informasi AWS DMS terperinci tentang Panduan AWS Database Migration Service Pengguna dan Referensi AWS Database Migration Service API. Secara khusus, Anda dapat mengetahui cara menyiapkan klaster Neptune sebagai target migrasi di Menggunakan Amazon Neptune sebagai Target untuk AWS Database Migration Service.

Berikut adalah beberapa prasyarat untuk mengimpor data ke Neptune menggunakan AWS DMS:

- Anda perlu membuat objek pemetaan AWS DMS tabel untuk menentukan bagaimana data harus diekstraksi dari database sumber (lihat Menentukan pemilihan tabel dan transformasi dengan pemetaan tabel menggunakan JSON di Userguide untuk detailnya). AWS DMS Objek konfigurasi pemetaan tabel ini menentukan tabel yang harus dibaca dan dalam urutan apa, dan bagaimana kolom mereka diberi nama. Hal ini juga dapat menyaring baris yang disalin dan memberikan transformasi nilai sederhana seperti mengonversi ke huruf kecil atau pembulatan.
- Anda akan perlu untuk membuat `GraphMappingConfig` Neptune untuk menentukan bagaimana data yang diekstrak dari database sumber harus dimuat ke Neptune. Untuk data RDF (kueri menggunakan SPARQL), `GraphMappingConfig` ditulis dalam bahasa pemetaan R2RML standar W3. Untuk data grafik properti (kueri menggunakan Gremlin), `GraphMappingConfig` adalah

objek JSON, dijelaskan dalam GraphMappingConfig Tata Letak untuk Property-Graph/Data Gremlin.

- Anda harus menggunakan AWS DMS untuk membuat instance replikasi di VPC yang sama dengan cluster DB Neptunus Anda, untuk memediasi transfer data.
- Anda juga akan membutuhkan bucket Amazon S3 untuk digunakan sebagai penyimpanan menengah untuk penahapan data migrasi.

Membuat Neptunus GraphMappingConfig

`GraphMappingConfig` yang Anda buat menentukan bagaimana data diekstrak dari penyimpanan sumber data harus dimuat ke dalam klaster DB Neptune. Formatnya berbeda tergantung apakah itu dimaksudkan untuk memuat data RDF atau untuk memuat data properti-grafik.

Untuk data RDF, Anda dapat menggunakan bahasa W3 R2RML untuk pemetaan data relasional ke RDF.

Jika Anda memuat data properti-grafik yang akan bertanya menggunakan Gremlin, Anda membuat objek JSON untuk `GraphMappingConfig`.

GraphMappingConfig Tata Letak untuk RDF/SPARQL Data

Jika Anda memuat data RDF yang akan dikueri menggunakan SPARQL, Anda menulis `GraphMappingConfig` di R2RML. `R2RML` adalah bahasa W3 standar untuk pemetaan data relasional untuk RDF. Inilah satu contohnya:

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/ns#> .

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "nodes" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{id}";
        rr:class ex:Employee;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "label" ];
    ] .
```

Inilah contoh lainnya:

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<#TriplesMap2>
    rr:logicalTable [ rr:tableName "Student" ];
    rr:subjectMap   [ rr:template "http://example.com/{ID}{Name}";
                      rr:class foaf:Person ];
    rr:predicateObjectMap [
        rr:predicate ex:id ;
        rr:objectMap  [ rr:column "ID";
                        rr:datatype xsd:integer ]
    ];
    rr:predicateObjectMap [
        rr:predicate foaf:name ;
        rr:objectMap  [ rr:column "Name" ]
    ] .
```

Rekomendasi W3 di R2RLL: Pemetaan Bahasa RDB ke RDF menyediakan rincian bahasa.

GraphMappingConfig Tata Letak untuk Property-Graph/Data Gremlin

`GraphMappingConfig` yang komparabel untuk data properti-grafik adalah objek JSON yang menyediakan aturan pemetaan untuk setiap entitas grafik yang akan dihasilkan dari data sumber. Templat berikut menunjukkan seperti apa penampakan setiap aturan dalam objek ini.

```
{
  "rules": [
    {
      "rule_id": "(an identifier for this rule)",
      "rule_name": "(a name for this rule)",
      "table_name": "(the name of the table or view being loaded)",
      "vertex_definitions": [
        {
          "vertex_id_template": "{col1}",
          "vertex_label": "(the vertex to create)",
          "vertex_definition_id": "(an identifier for this vertex)",
          "vertex_properties": [
            {
              "property_name": "(name of the property)",
              "property_value_template": "{col2} or text",
              "property_value_type": "(data type of the property)"
```

```
            }
          ]
        }
      ]
    },
    {
      "rule_id": "(an identifier for this rule)",
      "rule_name": "(a name for this rule)",
      "table_name": "(the name of the table or view being loaded)",
      "edge_definitions": [
        {
          "from_vertex": {
            "vertex_id_template": "{col1}",
            "vertex_definition_id": "(an identifier for the vertex referenced above)"
          },
          "to_vertex": {
            "vertex_id_template": "{col3}",
            "vertex_definition_id": "(an identifier for the vertex referenced above)"
          },
          "edge_id_template": {
            "label": "(the edge label to add)",
            "template": "{col1}_{col3}"
          },
          "edge_properties":[
            {
              "property_name": "(the property to add)",
              "property_value_template": "{col4} or text",
              "property_value_type": "(data type like String, int, double)"
            }
          ]
        }
      ]
    }
  ]
}
```

Perhatikan bahwa kehadiran label vertex menyiratkan bahwa vertex sedang dibuat di sini, sedangkan ketiadaannya menyiratkan bahwa vertex dibuat oleh sumber yang berbeda, dan definisi ini hanya menambahkan properti vertex.

Berikut adalah contoh aturan untuk catatan karyawan:

```
{
```

```
  "rules": [
    {
      "rule_id": "1",
      "rule_name": "vertex_mapping_rule_from_nodes",
      "table_name": "nodes",
      "vertex_definitions": [
        {
          "vertex_id_template": "{emp_id}",
          "vertex_label": "employee",
          "vertex_definition_id": "1",
          "vertex_properties": [
            {
              "property_name": "name",
              "property_value_template": "{emp_name}",
              "property_value_type": "String"
            }
          ]
        }
      ]
    },
    {
      "rule_id": "2",
      "rule_name": "edge_mapping_rule_from_emp",
      "table_name": "nodes",
      "edge_definitions": [
        {
          "from_vertex": {
            "vertex_id_template": "{emp_id}",
            "vertex_definition_id": "1"
          },
          "to_vertex": {
            "vertex_id_template": "{mgr_id}",
            "vertex_definition_id": "1"
          },
          "edge_id_template": {
            "label": "reportsTo",
            "template": "{emp_id}_{mgr_id}"
          },
          "edge_properties":[
            {
              "property_name": "team",
              "property_value_template": "{team}",
              "property_value_type": "String"
            }
```

```
          ]
        }
      ]
    }
  ]
}
```

Membuat tugas AWS DMS replikasi dengan Neptunus sebagai target

Setelah Anda membuat pemetaan tabel dan konfigurasi pemetaan grafik Anda, gunakan proses berikut untuk memuat data dari penyimpanan sumber ke Neptune. Konsultasikan AWS DMS dokumentasi untuk detail lebih lanjut tentang pertanyaan tersebut. APIs

Buat contoh AWS DMS replikasi

Buat instance AWS DMS replikasi di VPC tempat cluster DB Neptunus Anda berjalan (lihat Bekerja dengan Instans Replikasi DMS dan di AWS Panduan Pengguna). CreateReplicationInstance AWS DMS Anda dapat menggunakan AWS CLI perintah seperti berikut untuk melakukannya:

```
aws dms create-replication-instance \
    --replication-instance-identifier (the replication instance identifier) \
    --replication-instance-class (the size and capacity of the instance, like
 'dms.t2.medium') \
    --allocated-storage (the number of gigabytes to allocate for the instance
 initially) \
    --engine-version (the DMS engine version that the instance should use) \
    --vpc-security-group-ids (the security group to be used with the instance)
```

Buat AWS DMS titik akhir untuk database sumber

Langkah selanjutnya adalah membuat AWS DMS titik akhir untuk penyimpanan data sumber Anda. Anda dapat menggunakan AWS DMS CreateEndpointAPI AWS CLI seperti ini:

```
aws dms create-endpoint \
    --endpoint-identifier (source endpoint identifier) \
    --endpoint-type source \
    --engine-name (name of source database engine) \
    --username (user name for database login) \
    --password (password for login) \
    --server-name (name of the server) \
    --port (port number) \
```

```
    --database-name (database name)
```

Siapkan bucket Amazon S3 untuk Neptunus untuk digunakan untuk mementaskan data

Jika Anda tidak memiliki bucket Amazon S3 yang dapat Anda gunakan untuk penahapan data, buat satu bucket seperti yang dijelaskan di Membuat Bucket di Panduan Memulai Amazon S3, atau Bagaimana Cara Membuat Bucket S3? di Panduan Pengguna Konsol.

Anda harus membuat kebijakan IAM yang memberikan izin `GetObject`, `PutObject`, `DeleteObject` dan `ListObject` ke bucket jika Anda belum memilikinya:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket"
      ]
    },
    {
      "Sid": "AllObjectActions",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

Jika klaster DB Neptune Anda memiliki autentikasi IAM diaktifkan, Anda juga akan perlu untuk menyertakan kebijakan berikut:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:111122223333:cluster-resource-id/
*"
    }
  ]
}
```

Buat IAM role sebagai dokumen kepercayaan untuk dilampiri kebijakan:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "dms.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Sid": "neptune",
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Setelah melampirkan kebijakan ke peran, lampirkan peran ke klaster DB Neptune Anda. Ini akan memungkinkan AWS DMS untuk menggunakan bucket untuk mementaskan data yang sedang dimuat.

Buat titik akhir Amazon S3 di VPC Neptunus

Sekarang buat titik akhir VPC Gateway untuk bucket Amazon S3 perantara Anda, di VPC tempat klaster Neptune Anda berada. Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI untuk melakukan ini, seperti yang dijelaskan dalam Membuat titik akhir gateway.

Buat titik akhir AWS DMS target untuk Neptunus

Buat AWS DMS titik akhir untuk cluster DB Neptunus target Anda. Anda dapat menggunakan AWS DMS CreateEndpointAPI dengan `NeptuneSettings` parameter seperti ini::

```
aws dms create-endpoint \
    --endpoint-identifier (target endpoint identifier) \
    --endpoint-type target \
    --engine-name neptune \
    --server-name (name of the server) \
    --port (port number) \
    --neptune-settings '{ \
      "ServiceAccessRoleArn": "(ARN of the service access role)", \
      "S3BucketName": "(name of S3 bucket to use for staging files when migrating)", \
      "S3BucketFolder": "(name of the folder to use in that S3 bucket)", \
      "ErrorRetryDuration": (number of milliseconds to wait between bulk-load retries),
 \
      "MaxRetryCount": (the maximum number of times to retry a failing bulk-load job),
 \
      "MaxFileSize": (maximum file size, in bytes, of the staging files written to S3),
 \
```

```
    "IamAuthEnabled": (set to true if IAM authentication is enabled on the Neptune
cluster) }'
```

Objek JSON yang diteruskan ke AWS DMS `CreateEndpoint` API dalam `NeptuneSettings` parameternya memiliki bidang berikut:

- **`ServiceAccessRoleArn`** – (wajib) ARN dari IAM role yang memungkinkan akses berbutir halus ke bucket S3 yang digunakan untuk migrasi tahap data ke Neptune. Peran ini juga harus memiliki izin untuk mengakses klaster DB Neptune Anda jika otorisasi IAM diaktifkan di atasnya.
- **`S3BucketName`** – (wajib) Untuk migrasi Beban Penuh, instans replikasi mengonversi semua data RDS ke CSV, file quad dan unggah mereka ke bucket penahapan ini di S3 dan kemudian memuat massal mereka ke Neptune.
- **`S3BucketFolder`** – (wajib) Folder untuk digunakan dalam bucket penahapan S3.
- **`ErrorRetryDuration`** – (opsional) Jumlah milidetik untuk menunggu setelah permintaan Neptune gagal sebelum membuat permintaan coba lagi. Default-nya adalah 250.
- **`MaxRetryCount`**— (opsional) Jumlah maksimum permintaan coba ulang AWS DMS harus dilakukan setelah kegagalan yang dapat dicoba ulang. Default-nya adalah 5.
- **`MaxFileSize`** – (opsional) Ukuran maksimum dalam byte dari setiap file penahapan yang disimpan ke S3 selama migrasi. Default adalah 1.048.576 KB (1 GB).
- **`IsIAMAuthEnabled`** – (opsional) Atur ke `true` jika autentikasi IAM diaktifkan pada klaster DB Neptune, atau `false` jika tidak. Nilai default-nya `false`.

Uji koneksi ke titik akhir baru

Anda dapat menguji koneksi ke masing-masing titik akhir baru ini menggunakan AWS DMS TestConnectionAPI seperti ini:

```
aws dms test-connection \
    --replication-instance-arn (the ARN of the replication instance) \
    --endpoint-arn (the ARN of the endpoint you are testing)
```

Buat tugas AWS DMS replikasi

Setelah Anda berhasil menyelesaikan langkah-langkah sebelumnya, buat tugas replikasi untuk memigrasikan data dari penyimpanan data sumber Anda ke Neptunus, menggunakan API seperti ini: AWS DMS CreateReplicationTask

```
aws dms create-replication-task \
    --replication-task-identifier (name for the replication task) \
    --source-endpoint-arn (ARN of the source endpoint) \
    --target-endpoint-arn (ARN of the target endpoint) \
    --replication-instance-arn (ARN of the replication instance) \
    --migration-type full-load \
    --table-mappings (table-mapping JSON object or URI like 'file:///tmp/table-
mappings,json') \
    --task-data (a GraphMappingConfig object or URI like 'file:///tmp/graph-mapping-
config.json')
```

Parameter `TaskData` menyediakan GraphMappingConfig yang menentukan bagaimana data yang disalin harus disimpan di Neptune.

Mulai tugas AWS DMS replikasi

Sekarang Anda dapat memulai tugas replikasi:

```
aws dms start-replication-task
    --replication-task-arn (ARN of the replication task started in the previous step)
    --start-replication-task-type start-replication
```

Memuat data ke Amazon Neptunus menggunakan kueri

Neptunus mendukung penulisan data secara langsung melalui operasi bahasa kueri. Anda dapat menggunakan operasi tulis standar seperti `CREATE` dan `MERGE` di OpenCypher, di SPARQL, atau `mergeV()` dan `INSERT` di Gremlin untuk menambah atau memodifikasi data `mergeE()` dalam grafik Anda. Operasi ini cocok untuk pembaruan tambahan dan penulisan transaksional.

Untuk memuat data dari Amazon S3, gunakan Menggunakan pemuat massal Amazon Neptunus untuk menelan data untuk kumpulan data besar yang membutuhkan kinerja yang dioptimalkan. Untuk kumpulan data yang lebih kecil dalam satu atau beberapa file Amazon S3, Anda dapat menggunakan fungsi pemuatan berbasis kueri untuk membaca dan memproses data secara langsung dalam kueri Anda.

Fungsi pemuatan berbasis kueri berikut tersedia:

OpenCypher: neptune.read ()

`neptune.read()`Fungsi ini membaca file CSV atau Parket dari Amazon S3 dalam `CALL` subquery, memungkinkan Anda memproses dan memuat data pada waktu kueri.

```
CALL neptune.read({
  source: "s3://bucket/data.csv",
  format: "csv"
})
YIELD row
CREATE (n:Person {id: row.id, name: row.name})
```

Untuk dokumentasi lengkap, lihatneptune.read ().

SPARQL: MUAT dan BONGKAR

`LOAD`Operasi SPARQL mengimpor data RDF dari URI ke dalam grafik bernama. `UNLOAD`mengekspor data dari grafik ke Amazon S3.

```
LOAD <s3://bucket/data.ttl> INTO GRAPH <http://example.org/graph>
```

Untuk dokumentasi lengkap, lihatMenggunakan SPARQL UPDATE LOAD untuk mengimpor data ke Neptune.

Gremlin: io () langkah

Anda juga dapat menggunakan `g.io(URL).read()` langkah Gremlin untuk membaca dalam file data dalam GraphMl (format XHTML), GraphSon (format JSON), dan format lainnya.

```
g.io("s3://bucket/data.graphml").read().iterate()
```

Lihat TinkerPopdokumentasi untuk detailnya.

Penyimpanan

Neptunus mendukung pengumpulan sampah kamus (GC) untuk data grafik properti, yang dapat diaktifkan melalui parameter `neptune_lab_mode` `neptune_streams`saat tidak aktif. Saat diaktifkan, pekerjaan latar belakang ini membersihkan entri kamus yang tidak digunakan, berpotensi mengurangi laju pertumbuhan data. Fitur ini dapat berjalan dalam dua mode: soft_delete (menandai entri sebagai dihapus tanpa penghapusan eksplisit) dan diaktifkan (secara eksplisit menghapus entri). Proses GC dapat memengaruhi kinerja sistem dengan bersaing dengan utas kueri untuk sumber daya seperti CPU dan cache buffer, dan dapat berjalan dengan konkurensi maksimum 16 utas.

Neptunus juga mendukung IDs tepi yang dihasilkan server sebaris, yang dapat diaktifkan melalui parameter konfigurasi saat neptune_streams tidak aktif. Ketika fitur ini diaktifkan, server menghasilkan inline unik IDs untuk tepi yang tidak memiliki ID yang ditentukan pengguna, menggunakan awalan cadangan “neptune_reserved”. Inline ini tidak IDs disimpan dalam kamus, yang dapat meningkatkan efisiensi penyimpanan.

Topik

- Pengumpulan sampah kamus Neptunus
- ID tepi yang dihasilkan server sebaris Neptunus

Pengumpulan sampah kamus Neptunus

Neptunus mendukung pengumpulan sampah kamus (GC) yang dapat diaktifkan melalui parameter untuk data grafik `neptune_lab_mode` properti. Ini dapat diaktifkan untuk cluster dengan hanya data grafik properti ketika tidak `neptune_streams` diaktifkan. Fitur ini secara otomatis dinonaktifkan jika `neptune_streams` diaktifkan atau ada data yang belum kedaluwarsa`neptune_streams`. Fitur ini membutuhkan reboot instance penulis untuk mengaktifkan. Fitur ini tersedia dari rilis mesin 1.4.3.0.

Saat diaktifkan, entri kamus yang tidak digunakan dibersihkan oleh pekerjaan latar belakang. Itu tidak mengurangi`VolumeBytesUsed`, melainkan membebaskan ruang dalam indeks untuk sisipan baru. Tingkat pertumbuhan cenderung `VolumeBytesUsed` lebih kecil ketika kamus GC diaktifkan relatif terhadap saat tidak.

Pengumpulan sampah kamus berjalan di latar belakang dan memindai semua grafik dan data kamus untuk menemukan istilah yang tidak digunakan. Proses baru dipicu saat start up setelah sekitar 6%

dari data telah berubah. GC bersaing dengan utas kueri untuk sumber daya server seperti CPU, cache buffer, undo pembuatan log, dan I/O operasi tulis, yang berpotensi mengurangi throughput kueri. Karena GC memindai data yang tidak disentuh secara aktif oleh kueri, itu dapat memengaruhi cache buffer pada node penulis. Cluster dapat melihat I/O operasi penulisan tambahan dan memiliki lebih banyak log batal untuk dibersihkan saat GC melakukan penghapusan baru, yang juga dapat menghasilkan nilai yang lebih tinggi untuk metrik. `UndoLogListSize`

GC dapat dijalankan dalam dua mode, `soft_delete` dan`enabled`. Saat dijalankan dalam `soft_delete` mode, entri kamus yang tidak digunakan ditandai dihapus (soft_delete) tetapi tidak dihapus secara eksplisit. Mode ini juga dapat digunakan untuk memahami karakteristik kinerja setelah operasi latar belakang dihidupkan. Saat `enabled` mode digunakan, entri dihapus secara eksplisit (hapus 'keras'). Disarankan untuk menjalankan GC dalam `soft_delete` mode untuk jangka waktu tertentu sebelum beralih ke `enabled` mode.

Kamus GC mendukung konkurensi maksimum 16 (pada mesin dengan 16 atau lebih core). Ini berjalan secara default dengan satu utas, tetapi dapat dijalankan dengan konkurensi yang lebih tinggi saat diaktifkan untuk pertama kalinya. Thread GC Kamus berjalan pada prioritas yang sama dengan utas kueri, dan mereka bersaing dengan sumber daya pada penulis secara setara.

Kamus GC dapat diaktifkan melalui `neptune_lab_mode` parameter dengan mengatur `DictionaryGCMode` kunci. Ia menerima tiga nilai yang mungkin: `disabled` (default),`soft_delete`, atau`enabled`. Misalnya, contoh kode berikut akan mengatur `DictionaryGCMode` ke`soft_delete`:

```
neptune_lab_mode = 'DictionaryGCMode=soft_delete'
```

Parameter konkurensi,`DictionaryGCConcurrency`, adalah opsional dan dapat mengambil nilai antara 1 dan 16. Jika diatur ke nilai yang lebih tinggi dari minimum 16 dan jumlah core, konkurensi dibatasi pada nilai itu.

```
neptune_lab_mode = 'DictionaryGCMode=soft_delete,DictionaryGCConcurrency=2'
```

Pekerjaan kamus GC diaktifkan di latar belakang setelah server dimulai, setelah ada beberapa data yang tersedia. Status mesin menampilkan status kamus GC saat ini. Contoh output yang ditunjukkan di bawah ini menunjukkan bahwa kamus GC dalam `soft_delete` mode dan berjalan dengan konkurensi 2. Jika pekerjaan latar belakang berjalan, itu bisa secara aktif memindai entri kamus yang tidak digunakan dan menghapusnya, atau menunggu set penghapusan baru untuk memicu babak baru GC.

```
{"status":"healthy",...,"labMode":
{"ObjectIndex":"disabled","DictionaryGC":"{Mode=enabled,Concurrency=2}"},...}
```

Kamus GC dijeda ketika salah satu dari kondisi ini terpenuhi:

- Beban massal aktif.
- Memori yang dapat dibebaskan kurang dari 15Gb.
- `UndoLogListSize`lebih tinggi dari 1.000.000.

ID tepi yang dihasilkan server sebaris Neptunus

Neptunus mendukung Server-Generated Edge sebaris. IDs Ini dapat diaktifkan melalui `neptune_enable_inline_server_generated_edge_id` parameter konfigurasi Neptunus `neptune_streams` ketika tidak diaktifkan. Fitur ini tersedia untuk kueri Gremlin yang dimulai dengan rilis mesin 1.4.3.0, dan akan tersedia untuk kueri OpenCypher di rilis mendatang.

Edge ID adalah pengidentifikasi unik untuk sebuah edge. ID tepi dapat diberikan saat memasukkan tepi. Jika tidak ada ID yang disediakan, server menghasilkan dan menetapkan ID berbasis UUID ke edge secara default. Seperti ID yang ditentukan pengguna, ID yang dihasilkan server berbasis UUID disimpan dalam kamus.

Ketika `neptune_enable_inline_server_generated_edge_id` fitur diaktifkan, server menghasilkan ID inline yang unik ketika tidak ada ID yang disediakan dalam kueri. Tepi sebaris tidak IDs disimpan dalam kamus, meningkatkan efisiensi penyimpanan. Inline yang dihasilkan server IDs dimulai dengan awalan cadangan. `neptune_reserved`

> **Warning**
>
> Neptunus mencadangkan awalan untuk server `'neptune_reserved'` yang dihasilkan sebaris. IDs Kesalahan akan ditampilkan untuk kueri yang mencoba menyisipkan data dengan ID yang ditentukan pengguna yang dimulai dengan awalan cadangan.

Fitur ID tepi yang dihasilkan server sebaris dapat diaktifkan dengan menyetel parameter tingkat cluster ke. `neptune_enable_inline_server_generated_edge_id` 1 Diperlukan reboot instance. Contoh berikut memungkinkan fitur ID tepi yang dihasilkan server:

```
"ParameterName=neptune_enable_inline_server_generated_edge_id,ParameterValue=1,ApplyMethod=pend
reboot"
```

Untuk memverifikasi apakah fitur diaktifkan, Anda dapat memeriksa fitur dalam status mesin. Fitur ini secara otomatis dinonaktifkan jika `neptune_streams` diaktifkan. Contoh keluaran berikut menunjukkan status mesin untuk fitur yang diaktifkan:

```
"features":{"InlineServerGeneratedEdgeId":"enabled"}
```

Contoh Gremlin berikut menambahkan tepi tanpa ID yang ditentukan pengguna saat fitur ID tepi yang dihasilkan server sebaris diaktifkan:

```
curl - X POST--url https: //<neptune-cluster-endpoint>:8182/gremlin/ --data
 '{"gremlin":"g.withSideEffect(\"Neptune#disablePushdownOptimization\",
 true).addV().property(id, \"a\").addV().property(id, \"b\").addE(\"el\").to(V(\"a
\"))"}'
{
    "requestId": "b6b84605-53ad-4c04-baf1-7f0f31a3aeaf",
    "status": {
        "message": "",
        "code": 200,
        "attributes": {
            "@type": "g:Map",
            "@value": []
        }
    },
    "result": {
        "data": {
            "@type": "g:List",
            "@value": [{
                "@type": "g:Edge",
                "@value": {
                    "id": "neptune_reserved_231850767",
                    "label": "el",
                    "inVLabel": "vertex",
                    "outVLabel": "vertex",
                    "inV": "a",
                    "outV": "b"
                }
            }]
        },
        "meta": {
```

```
            "@type": "g:Map",
            "@value": []
        }
    }
}
```

Mengajukan Kueri Grafik Neptune

Neptune mendukung bahasa kueri grafik berikut untuk mengakses grafik:

- Gremlin, didefinisikan oleh Apache TinkerPop untuk membuat dan menanyakan grafik properti.

  Sebuah kueri di Gremlin adalah sebuah traversal yang terdiri dari langkah-langkah berlainan, yang masing-masing mengikuti edge ke simpul.

  Lihat Mengakses grafik Neptune dengan Gremlin untuk mempelajari cara menggunakan Gremlin di Neptune, dan Kepatuhan standar Gremlin di Amazon Neptune untuk menemukan detail spesifik tentang implementasi Neptune Gremlin.
- OpenCypheradalah bahasa kueri deklaratif untuk grafik properti yang awalnya dikembangkan oleh Neo4j, kemudian bersumber terbuka pada tahun 2015, dan berkontribusi pada proyek OpenCypher di bawah lisensi open-source Apache 2. Sintaksnya didokumentasikan dalam spesifikasi OpenCypher.
- SPARQL adalah bahasa deklaratif berdasarkan grafik pola-pencocokan, untuk mengajukan kueri data RDF. Hal ini didukung oleh World Wide Web Consortium.

  Lihat Mengakses grafik Neptune dengan SPARQL untuk mempelajari cara menggunakan SPARQL di Neptune, dan Kepatuhan standar SPARQL di Amazon Neptunus untuk menemukan detail spesifik tentang implementasi Neptune SPARQL.

> **Note**
>
> Baik Gremlin dan OpenCypher dapat digunakan untuk menanyakan data grafik properti apa pun yang disimpan di Neptunus, terlepas dari bagaimana itu dimuat.

Topik

- Antrean kueri di Amazon Neptune
- Cache paket kueri di Amazon Neptunus
- Menyuntikkan ID Kustom Ke Dalam Gremlin Neptune atau Kueri SPARQL
- Mengakses grafik Neptune dengan Gremlin
- Mengakses Grafik Neptunus dengan OpenCypher

- Mengakses grafik Neptune dengan SPARQL

Antrean kueri di Amazon Neptune

Ketika mengembangkan dan menyesuaikan aplikasi grafik, mengetahui implikasi dari bagaimana kueri diantrekan oleh database akan sangat membantu. Di Amazon Neptune, antrean kueri terjadi sebagai berikut:

- Jumlah maksimum kueri yang dapat diantrekan per instans, terlepas dari ukuran instans, adalah 8.192. Setiap kueri yang melebihi jumlah tersebut ditolak dan gagal dengan `ThrottlingException`.
- Jumlah maksimum kueri yang dapat dijalankan pada satu waktu ditentukan oleh jumlah thread pekerja yang ditetapkan, yang umumnya diatur ke dua kali jumlah core CPU virtual (vCPUs) yang tersedia.
- Latensi kueri menyertakan waktu yang dihabiskan sebuah kueri dalam antrean serta perjalanan memutari jaringan dan waktu yang benar-benar diperlukan kueri untuk dieksekusi.

Menentukan berapa banyak kueri dalam antrean Anda pada saat tertentu

`MainRequestQueuePendingRequests` CloudWatch Metrik mencatat jumlah permintaan yang menunggu dalam antrian input dengan perincian lima menit (lihat). Metrik Neptunus CloudWatch

Untuk Gremlin, Anda dapat memperoleh jumlah kueri dalam antrean menggunakan nilai `acceptedQueryCount` yang dikembalikan oleh API status kueri Gremlin. Namun, perlu diingat bahwa nilai `acceptedQueryCount` yang dikembalikan oleh API status kueri SPARQL mencakup semua kueri yang diterima sejak server dimulai, termasuk kueri yang telah selesai.

Bagaimana antrean kueri dapat memengaruhi waktu habis

Seperti yang dinyatakan di atas, latensi kueri menyertakan waktu yang dihabiskan sebuah kueri dalam antrean serta waktu yang diperlukan kueri untuk dieksekusi.

Karena periode waktu habis kueri umumnya diukur mulai dari ketika memasuki antrean, antrean yang bergerak lambat dapat membuat banyak kueri kehabisan waktu segera setelah keluar dari antrean. Hal ini jelas tidak diinginkan, jadi hindari antrean kueri dalam jumlah besar kecuali kueri tersebut dapat dieksekusi dengan cepat.

Cache paket kueri di Amazon Neptunus

Ketika kueri dikirimkan ke Neptunus, string kueri diurai, dioptimalkan, dan diubah menjadi rencana kueri, yang kemudian dieksekusi oleh mesin. Aplikasi sering didukung oleh pola kueri umum yang dipakai dengan nilai yang berbeda. Cache rencana kueri dapat mengurangi latensi keseluruhan dengan menyimpan paket kueri dan dengan demikian menghindari penguraian dan pengoptimalan untuk pola berulang tersebut.

Query Plan Cache dapat digunakan untuk OpenCypherkueri — baik kueri non-parameter maupun parameter. Ini diaktifkan untuk READ, dan untuk HTTP dan Bolt. Ini tidak didukung untuk kueri mutasi OC. Ini tidak didukung untuk kueri Gremlin atau SPARQL.

Cara memaksa mengaktifkan atau menonaktifkan cache rencana kueri

Cache paket kueri diaktifkan secara default untuk kueri berparameter latensi rendah. Rencana untuk kueri berparameter hanya di-cache ketika latensi lebih rendah dari ambang batas 100 ms. Perilaku ini dapat diganti berdasarkan per-kueri (diparameterisasi atau tidak) oleh Petunjuk Kueri tingkat kueri. `QUERY:PLANCACHE` Itu perlu digunakan dengan `USING` klausa. Petunjuk kueri menerima `enabled` atau `disabled` sebagai nilai.

```
# Forcing plan to be cached or reused
% curl -k https://<endpoint>:<port>/opencypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1"

% curl -k https://<endpoint>:<port>/opencypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" RETURN \$arg" \
  -d "parameters={\"arg\": 123}"

# Forcing plan to be neither cached nor reused
% curl -k https://<endpoint>:<port>/opencypher \
  -d "query=Using QUERY:PLANCACHE \"disabled\" MATCH(n) RETURN n LIMIT 1"
```

Cara menentukan apakah rencana di-cache atau tidak

Untuk HTTP READ, jika kueri dikirimkan dan paket di-cache, `explain` akan menampilkan detail yang relevan dengan cache rencana kueri.

```
% curl -k https://<endpoint>:<port>/opencypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1" \
```

```
  -d "explain=[static|details]"

Query: <QUERY STRING>
Plan cached by request: <REQUEST ID OF FIRST TIME EXECUTION>
Plan cached at: <TIMESTAMP OF FIRST TIME EXECUTION>
Parameters: <PARAMETERS, IF QUERY IS PARAMETERIZED QUERY>
Plan cache hits: <NUMBER OF CACHE HITS FOR CACHED PLAN>
First query evaluation time: <LATENCY OF FIRST TIME EXECUTION>

The query has been executed based on a cached query plan. Detailed explain with
 operator runtime statistics can be obtained by running the query with plan cache
 disabled (using HTTP parameter planCache=disabled).
```

Saat menggunakan Bolt, fitur jelaskan tidak didukung.

Penggusuran

Rencana kueri diusir oleh cache time to live (TTL) atau ketika jumlah maksimum paket kueri cache telah tercapai. Saat paket kueri terpukul, TTL disegarkan. Defaultnya adalah:

- 1000 - Jumlah maksimum paket yang dapat di-cache per instance.
- TTL - 300.000 milidetik atau 5 menit. Cache hit memulai ulang TTL, dan mengatur ulang kembali ke 5 menit.

Kondisi yang menyebabkan rencana tidak di-cache

Cache paket kueri tidak akan digunakan dalam kondisi berikut:

1. Saat kueri dikirimkan menggunakan petunjuk `QUERY:PLANCACHE "disabled"` kueri. Anda dapat menjalankan kembali kueri dan menghapus `QUERY:PLANCACHE "disabled"` untuk mengaktifkan cache rencana kueri.
2. Jika kueri yang dikirimkan bukan kueri berparameter dan tidak berisi petunjuk. `QUERY:PLANCACHE "enabled"`
3. Jika waktu evaluasi kueri lebih besar dari ambang latensi, kueri tidak di-cache dan dianggap sebagai kueri yang berjalan lama yang tidak akan mendapat manfaat dari cache paket kueri.
4. Jika kueri berisi pola yang tidak mengembalikan hasil apa pun.
   - yaitu `MATCH (n:nonexistentLabel) return n` ketika ada nol node dengan label yang ditentukan.

- yaitu `MATCH (n {name: $param}) return n` dengan `parameters={"param": "abcde"}` ketika ada nol node yang mengandung`name=abcde`.

5. Jika parameter kueri adalah tipe komposit, seperti a `list` atau `amap`.

```
curl -k https://<endpoint>:<port>/opencypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" RETURN \$arg" \
  -d "parameters={\"arg\": [1, 2, 3]}"

curl -k https://<endpoint>:<port>/opencypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" RETURN \$arg" \
  -d "parameters={\"arg\": {\"a\": 1}}"
```

6. Jika parameter query adalah string yang belum menjadi bagian dari beban data atau operasi penyisipan data. Misalnya, jika `CREATE (n {name: "X"})` dijalankan untuk menyisipkan`"X"`, `RETURN "X"` maka di-cache, sementara tidak `RETURN "Y"` akan di-cache, karena `"Y"` belum dimasukkan dan tidak ada dalam database.

Menyuntikkan ID Kustom Ke Dalam Gremlin Neptune atau Kueri SPARQL

Secara default, Neptune memberikan nilai `queryId` yang unik untuk setiap kueri. Anda dapat menggunakan ID ini untuk mendapatkan informasi tentang kueri yang berjalan (lihat API status kueri Gremlin atau API status kueri SPARQL), atau membatalkannya (lihat Pembatalan kueri Gremlin atau Pembatalan kueri SPARQL).

Neptune juga memungkinkan Anda menentukan nilai `queryId` sendiri untuk kueri Gremlin atau SPARQL, baik di header HTTP, atau untuk kueri SPARQL menggunakan petunjuk kueri `queryId`. Menugaskan `queryID` Anda sendiri memudahkan pelacakan kueri untuk mendapatkan statusnya atau membatalkannya.

Menyuntikkan Nilai **`queryId`** Kustom Menggunakan Header HTTP

Untuk Gremlin dan SPARQL, header HTTP dapat digunakan untuk menyuntikkan nilai `queryId` Anda sendiri ke dalam kueri.

Contoh Gremlin

```
curl -XPOST https://your-neptune-endpoint:port \
```

```
    -d "{\"gremlin\": \
        \"g.V().limit(1).count()\" , \
        \"queryId\":\"4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47\"  }"
```

Contoh SPARQL

```
curl https://your-neptune-endpoint:port/sparql \
    -d "query=SELECT * WHERE { ?s ?p ?o } " \
       --data-urlencode \
       "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Menyuntikkan Nilai **`queryId`** Kustom menggunakan Petunjuk Kueri SPARQL

Berikut adalah contoh bagaimana Anda akan menggunakan petunjuk kueri `queryId` SPARQL untuk menyuntikkan nilai `queryId` kustom ke dalam kueri SPARQL:

```
curl https://your-neptune-endpoint:port/sparql \
    -d "query=PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#> \
       SELECT * WHERE { hint:Query hint:queryId \"4d5c4fae-
aa30-41cf-9e1f-91e6b7dd6f47\" \
       {?s ?p ?o}}"
```

Menggunakan Nilai **`queryId`** untuk Memeriksa Status Kueri

Contoh Gremlin

```
curl https://your-neptune-endpoint:port/gremlin/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Contoh SPARQL

```
curl https://your-neptune-endpoint:port/sparql/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Mengakses grafik Neptune dengan Gremlin

Amazon Neptunus kompatibel dengan TinkerPop Apache dan Gremlin. Ini berarti Anda dapat terhubung ke instans DB Neptunus dan menggunakan bahasa traversal Gremlin untuk menanyakan

grafik (lihat Grafik dalam dokumentasi Apache). TinkerPop Untuk perbedaan dalam implementasi Neptune Gremlin, lihat Kepatuhan standar Gremlin.

Sebuah traversal di Gremlin adalah serangkaian langkah berantai. Traversal dimulai pada sebuah vertex (atau edge). Traversal menelusuri grafik dengan mengikuti edge keluar dari setiap vertex lalu edge keluar dari vertex tersebut. Setiap langkah adalah operasi dalam traversal. Untuk informasi selengkapnya, lihat The Traversal dalam TinkerPop dokumentasi.

Versi mesin Neptune yang berbeda mendukung versi Gremlin yang berbeda. Periksa halaman rilis mesin versi Neptunus yang Anda jalankan untuk menentukan rilis Gremlin mana yang didukungnya atau lihat tabel berikut yang mencantumkan versi paling awal dan terbaru yang TinkerPop didukung oleh versi mesin Neptunus yang berbeda:

Versi Mesin Neptune	TinkerPop Versi Minimum	TinkerPop Versi Maksimum
`1.3.2.0 and newer`	`3.7.1`	`3.7.3`
`1.3.1.0`	`3.6.2`	`3.6.5`
`1.3.0.0`	`3.6.2`	`3.6.4`
`1.2.1.0 <= 1.2.1.2`	`3.6.2`	`3.6.2`
`1.1.1.0 <= 1.2.0.2`	`3.5.5`	`3.5.6`
`1.1.0.0 and older`	`(deprecated)`	`(deprecated)`

TinkerPop klien biasanya kompatibel ke belakang dalam seri (`3.6.x`, misalnya, atau`3.7.x`) dan sementara mereka sering dapat bekerja melintasi batas-batas tersebut, tabel di atas merekomendasikan kombinasi versi untuk digunakan untuk pengalaman dan kompatibilitas terbaik. Kecuali disarankan lain, umumnya yang terbaik adalah mematuhi pedoman ini dan meningkatkan aplikasi klien agar sesuai dengan versi yang TinkerPop Anda gunakan.

Saat memutakhirkan TinkerPop versi, selalu penting untuk merujuk ke TinkerPopdokumentasi pemutakhiran yang akan membantu Anda mengidentifikasi fitur baru yang dapat Anda manfaatkan, tetapi juga masalah yang mungkin perlu Anda waspadai saat mendekati peningkatan. Anda biasanya mengharapkan kueri dan fitur yang ada berfungsi setelah peningkatan kecuali sesuatu secara khusus disebut sebagai masalah untuk dipertimbangkan. Terakhir, penting untuk dicatat bahwa jika versi

yang Anda tingkatkan untuk memiliki fitur baru, Anda mungkin tidak dapat menggunakannya jika berasal dari versi yang lebih lambat dari yang didukung Neptunus.

Ada beberapa varian bahasa Gremlin dan dukungan untuk akses Gremlin dalam berbagai bahasa pemrograman. Untuk informasi selengkapnya, lihat Tentang Varian Bahasa Gremlin dalam dokumentasi. TinkerPop

Dokumentasi ini menjelaskan cara mengakses Neptunus dengan varian dan bahasa pemrograman berikut:

- Mengatur konsol Gremlin agar terhubung ke instans DB Neptune
- Menggunakan titik akhir HTTPS REST untuk menyambung ke instans DB Neptune
- Klien Gremlin berbasis Java untuk digunakan dengan Amazon Neptunus
- Menggunakan Python untuk terhubung ke instans DB Neptune
- Gunakan .NET untuk terhubung ke instans DB Neptune
- Gunakan Node.js untuk terhubung ke instans DB Neptune
- Menggunakan Go untuk terhubung ke instans DB Neptunus

Seperti dibahas dalamMengenkripsi koneksi ke database Amazon Neptunus Anda dengan SSL/HTTPS, Anda harus menggunakan Transport Layer Security/Secure Sockets Layer (TLS/SSL) saat menghubungkan ke Neptunus di semua Wilayah. AWS

Sebelum Anda mulai, Anda harus memiliki yang berikut:

- Instans DB Neptune. Untuk informasi tentang membuat instans DB Neptune, lihat Membuat cluster Amazon Neptunus.
- Instans Amazon EC2 di virtual private cloud (VPC) yang sama seperti instans DB Neptune Anda.

Untuk informasi lebih lanjut tentang memuat data ke dalam Neptune, termasuk prasyarat, format pemuatan, dan parameter beban, lihat Memuat data ke Amazon Neptune.

Topik

- Mengatur konsol Gremlin agar terhubung ke instans DB Neptune
- Menggunakan titik akhir HTTPS REST untuk menyambung ke instans DB Neptune
- Klien Gremlin berbasis Java untuk digunakan dengan Amazon Neptunus
- Menggunakan Python untuk terhubung ke instans DB Neptune

- Gunakan .NET untuk terhubung ke instans DB Neptune
- Gunakan Node.js untuk terhubung ke instans DB Neptune
- Menggunakan Go untuk terhubung ke instans DB Neptunus
- Menggunakan AWS SDK untuk menjalankan kueri Gremlin
- Petunjuk kueri Gremlin
- API status kueri Gremlin
- Pembatalan kueri Gremlin
- Support untuk sesi berbasis skrip Gremlin
- Transaksi Gremlin di Neptunus
- Menggunakan API Gremlin dengan Amazon Neptune
- Hasil kueri cache di Amazon Neptunus Gremlin
- Membuat peningkatan yang efisien dengan mergeV() Gremlin dan langkah-langkah mergeE()
- Membuat peningkatan Gremlin yang efisien dengan fold()/coalesce()/unfold()
- Menganalisis eksekusi kueri Neptune menggunakan explain Gremlin
- Menggunakan Gremlin dengan mesin kueri Neptunus DFE

Mengatur konsol Gremlin agar terhubung ke instans DB Neptune

Konsol Gremlin memungkinkan Anda bereksperimen dengan TinkerPop grafik dan kueri di lingkungan REPL (loop). read-eval-print

Menginstal konsol Gremlin dan menghubungkannya dengan cara biasa

Anda dapat menggunakan Konsol Gremlin untuk terhubung ke basis data grafik jarak jauh. Bagian berikut memandu Anda melalui penginstalan dan konfigurasi Konsol Gremlin untuk menghubungkan ke instans DB Neptune secara jarak jauh. Anda harus mengikuti petunjuk ini dari instans Amazon EC2 di virtual private cloud (VPC) yang sama seperti instans DB Neptune Anda.

Untuk bantuan menghubungkan ke Neptunus SSL/TLS dengan (yang diperlukan), lihat. Konfigurasi SSL/TLS

> **Note**
>
> Jika autentikasi IAM diaktifkan di cluster DB Neptunus Anda, ikuti petunjuk Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan konsol Gremlin untuk menginstal konsol Gremlin daripada instruksi di sini.

Untuk memasang Konsol Gremlin dan menyambung ke Neptune

1. Biner Konsol Gremlin memerlukan Java 8 atau Java 11. Instruksi ini mengasumsikan penggunaan Java 11. Anda dapat menginstal Java 11 pada instans EC2 Anda sebagai berikut:

   - Jika Anda menggunakan Amazon Linux 2 (AL2):

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

   - Jika Anda menggunakan Amazon Linux 2023 (AL2023):

     ```
     sudo yum install java-11-amazon-corretto-devel
     ```

   - Untuk distribusi lain, gunakan salah satu dari berikut ini yang sesuai:

     ```
     sudo yum install java-11-openjdk-devel
     ```

     atau:

     ```
     sudo apt-get install openjdk-11-jdk
     ```

2. Masukkan yang berikut ini untuk mengatur Java 11 sebagai runtime default pada instans EC2 Anda.

   ```
   sudo /usr/sbin/alternatives --config java
   ```

   Saat diminta, masukkan nomor untuk Java 11.

3. Unduh versi Konsol Gremlin yang sesuai dari situs web Apache. Anda dapat memeriksa versi Gremlin mana yang didukung versi Neptunus Anda. Mengakses grafik Neptune dengan Gremlin Misalnya, jika Anda memerlukan versi 3.7.2, Anda dapat mengunduh konsol Gremlin dari situs web Apache Tinkerpop ke instans EC2 Anda seperti ini:

```
wget https://archive.apache.org/dist/tinkerpop/3.7.2/apache-tinkerpop-gremlin-
console-3.7.2-bin.zip
```

4. Unzip file Konsol Gremlin.

```
unzip apache-tinkerpop-gremlin-console-3.7.2-bin.zip
```

5. Ubah direktori ke dalam direktori yang di-unzip.

```
cd apache-tinkerpop-gremlin-console-3.7.2
```

6. Di sudirektori `conf` dari direktori yang diekstrak, buat sebuah file bernama `neptune-remote.yaml` dengan teks berikut. Ganti *`your-neptune-endpoint`* dengan nama host atau alamat IP instans DB Neptunus Anda. Tanda kurung persegi (`[ ]`) wajib diisi.

> **Note**
>
> Untuk informasi tentang menemukan nama host instans DB Neptune Anda, lihat bagian Menghubungkan ke Titik Akhir Amazon Neptune..

```
hosts: [your-neptune-endpoint]
port: 8182
connectionPool: { enableSsl: true }
serializer: { className:
 org.apache.tinkerpop.gremlin.util.ser.GraphBinaryMessageSerializerV1,
              config: { serializeResultToString: true }}
```

> **Note**
>
> Serializer dipindahkan dari `gremlin-driver` modul ke `gremlin-util` modul baru di versi 3.7.0. Paket diubah dari org.apache.tinkerpop.gremlin.driver.ser menjadi org.apache.tinkerpop.gremlin.util.ser.

7. Di terminal, bernavigasilah ke direktori Konsol Gremlin (`apache-tinkerpop-gremlin-console-3.7.2`), lalu masukkan perintah berikut untuk menjalankan Konsol Gremlin.

```
bin/gremlin.sh
```

Anda akan melihat output berikut:

```
        \,,,/
        (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin>
```

Anda sekarang berada di prompt `gremlin>`. Anda akan memasukkan langkah-langkah yang tersisa pada prompt ini.

8. Di prompt `gremlin>`, masukkan hal berikut untuk menyambung ke instans DB Neptune.

```
:remote connect tinkerpop.server conf/neptune-remote.yaml
```

9. Di prompt `gremlin>`, masukkan hal berikut ini untuk beralih ke mode jarak jauh. Ini mengirimkan semua kueri Gremlin ke koneksi remote.

```
:remote console
```

10. Masukkan hal berikut untuk mengirim kueri ke Gremlin Graph.

```
g.V().limit(1)
```

11. Setelah Anda selesai, masukkan hal berikut untuk keluar dari Konsol Gremlin.

```
:exit
```

> **Note**
>
> Gunakan titik koma (`;`) atau karakter baris baru (`\n`) untuk memisahkan setiap pernyataan. Setiap traversal sebelum traversal akhir harus diakhiri dengan `next()` yang akan dieksekusi. Hanya data dari traversal akhir yang dikembalikan.

Untuk informasi lebih lanjut tentang implementasi Neptune dari Gremlin, lihat the section called “Kepatuhan standar Gremlin”.

Cara alternatif untuk terhubung ke konsol Gremlin

Kelemahan dari pendekatan koneksi normal

Cara paling umum untuk terhubung ke konsol Gremlin adalah yang dijelaskan di atas, menggunakan perintah seperti ini pada `gremlin>` prompt:

```
gremlin> :remote connect tinkerpop.server conf/(file name).yaml
gremlin> :remote console
```

Ini berfungsi dengan baik, dan memungkinkan Anda mengirim kueri ke Neptunus. Namun, itu mengeluarkan mesin skrip Groovy dari loop, jadi Neptunus memperlakukan semua kueri sebagai Gremlin murni. Ini berarti bahwa formulir kueri berikut gagal:

```
gremlin> 1 + 1
gremlin> x = g.V().count()
```

Yang paling dekat yang bisa Anda dapatkan menggunakan variabel saat terhubung dengan cara ini adalah dengan menggunakan `result` variabel yang dikelola oleh konsol dan mengirim kueri menggunakan `:>`, seperti ini:

```
gremlin> :remote console
==>All scripts will now be evaluated locally - type ':remote console' to return
 to remote mode for Gremlin Server - [krl-1-cluster.cluster-ro-cm9t6tfwbtsr.us-
east-1.neptune.amazonaws.com/172.31.19.217:8182]
gremlin> :> g.V().count()
==>4249

gremlin> println(result)
[result{object=4249 class=java.lang.Long}]

gremlin> println(result['object'])
[4249]
```

Cara berbeda untuk terhubung

Anda juga dapat terhubung ke konsol Gremlin dengan cara yang berbeda, yang mungkin Anda temukan lebih bagus, seperti ini:

```
gremlin> g = traversal().withRemote('conf/neptune.properties')
```

Di sini `neptune.properties` mengambil formulir ini:

```
gremlin.remote.remoteConnectionClass=org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteCon
gremlin.remote.driver.clusterFile=conf/my-cluster.yaml
gremlin.remote.driver.sourceName=g
```

`my-cluster.yaml`File akan terlihat seperti ini:

```
hosts: [my-cluster-abcdefghijk.us-east-1.neptune.amazonaws.com]
port: 8182
serializer: { className:
 org.apache.tinkerpop.gremlin.util.ser.GraphBinaryMessageSerializerV1,
              config: { serializeResultToString: false } }
connectionPool: { enableSsl: true }
```

> **Note**
>
> Serializer dipindahkan dari `gremlin-driver` modul ke `gremlin-util` modul baru di versi 3.7.0. Paket diubah dari org.apache.tinkerpop.gremlin.driver.ser menjadi org.apache.tinkerpop.gremlin.util.ser.

Mengkonfigurasi koneksi konsol Gremlin seperti itu memungkinkan Anda membuat jenis kueri berikut dengan sukses:

```
gremlin> 1+1
==>2

gremlin> x=g.V().count().next()
==>4249

gremlin> println("The answer was ${x}")
The answer was 4249
```

Anda dapat menghindari menampilkan hasilnya, seperti ini:

```
gremlin> x=g.V().count().next();[]
```

```
gremlin> println(x)
4249
```

Semua cara kueri yang biasa (tanpa langkah terminal) terus berfungsi. Contoh:

```
gremlin> g.V().count()
==>4249
```

Anda bahkan dapat menggunakan `g.io().read()`langkah untuk memuat file dengan koneksi semacam ini.

Autentikasi IAM

Neptunus mendukung otentikasi IAM untuk mengontrol akses ke cluster DB Anda. Jika autentikasi IAM diaktifkan, Anda harus menggunakan penandatanganan Signature Version 4 untuk mengautentikasi permintaan Anda. Untuk instruksi terperinci dan contoh kode untuk menghubungkan dari konsol Gremlin, lihat. Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan konsol Gremlin

Menggunakan titik akhir HTTPS REST untuk menyambung ke instans DB Neptune

Amazon Neptune menyediakan titik akhir HTTPS untuk kueri Gremlin. Antarmuka REST kompatibel dengan apa pun versi Gremlin yang digunakan klaster DB Anda (lihat halaman rilis mesin dari versi mesin Neptune yang Anda jalankan untuk menentukan rilis Gremlin yang didukungnya).

> **Note**
>
> Sebagaimana dibahas dalam Mengenkripsi koneksi ke database Amazon Neptunus Anda dengan SSL/HTTPS, Neptune sekarang mengharuskan Anda terhubung menggunakan HTTPS dan bukan HTTP. Selain itu, Neptunus saat ini tidak mendukung HTTP/2 untuk permintaan REST API. Klien harus menggunakan HTTP/1.1 saat menghubungkan ke titik akhir.

Petunjuk berikut memandu Anda menyambung ke titik akhir Gremlin menggunakan perintah `curl` dan HTTPS. Anda harus mengikuti petunjuk ini dari instans Amazon EC2 di virtual private cloud (VPC) yang sama seperti instans DB Neptune Anda.

Titik akhir HTTPS untuk kueri Gremlin ke instans DB Neptune adalah `https://your-neptune-endpoint:port/gremlin`.

> **Note**
>
> Untuk informasi tentang menemukan nama host instans DB Neptune Anda, lihat Menghubungkan ke Titik Akhir Amazon Neptune..

Untuk menyambung ke Neptune menggunakan titik akhir HTTP REST

Contoh berikut menggunakan curl untuk mengirimkan kueri Gremlin melalui POST HTTP. Kueri dikirimkan dalam format JSON dalam tubuh posting sebagai properti `gremlin`.

```
curl -X POST -d '{"gremlin":"g.V().limit(1)"}' https://your-neptune-endpoint:port/
gremlin
```

Contoh ini mengembalikan vertex pertama dalam grafik menggunakan traversal `g.V().limit(1)`. Anda dapat mengajukan kueri untuk sesuatu yang lain dengan menggantinya dengan traversal Gremlin lain.

> **Important**
>
> Secara default, titik akhir REST mengembalikan semua hasil dalam satu set hasil JSON. Jika set hasil ini terlalu besar, pengecualian `OutOfMemoryError` dapat terjadi pada instans DB Neptune.
>
> Anda dapat menghindari hal ini dengan mengaktifkan respons bongkahan (hasil dikembalikan dalam serangkaian respons terpisah). Lihat Gunakan header jejak HTTP opsional untuk mengaktifkan respons multi-bagian Gremlin.

Meskipun permintaan POST HTTP direkomendasikan untuk mengirim kueri Gremlin, tetapi memungkinkan juga untuk menggunakan permintaan GET HTTP:

```
curl -G "https://your-neptune-endpoint:port?gremlin=g.V().count()"
```

> **Note**
>
> Neptune tidak mendukung properti `bindings`.

Gunakan header jejak HTTP opsional untuk mengaktifkan respons multi-bagian Gremlin

Secara default, respons HTTP untuk kueri Gremlin dikembalikan dalam satu set hasil JSON. Dalam kasus set hasil yang sangat besar, ini dapat menyebabkan pengecualian `OutOfMemoryError` pada instans DB.

Namun, Anda dapat mengaktifkan respons bongkahan (respons yang dikembalikan dalam beberapa bagian yang terpisah). Anda melakukan ini dengan menyertakan header jejak transfer-encoding (TE) (`te: trailers`) dalam permintaan Anda. Lihat halaman MDN tentang header permintaan TE) untuk informasi lebih lanjut tentang header TE.

Ketika respons dikembalikan dalam beberapa bagian, akan sulit untuk mendiagnosis masalah yang terjadi setelah bagian pertama diterima, karena bagian pertama tiba dengan kode status HTTP `200` (OK). Kegagalan berikutnya biasanya menghasilkan badan pesan yang berisi respons yang rusak, pada akhirnya Neptune menambahkan pesan kesalahan.

Agar deteksi dan diagnosis kegagalan semacam ini jadi lebih mudah, Neptune juga mencakup dua bidang header baru dalam trailing header dari setiap bongkahan respons:

- `X-Neptune-Status` – berisi kode respons diikuti dengan nama pendek. Misalnya, dalam kasus keberhasilan, trailing header akan berupa: `X-Neptune-Status: 200 OK`. Dalam kasus kegagalan, kode respon akan menjadi salah satu kode kesalahan mesin Neptune, seperti `X-Neptune-Status: 500 TimeLimitExceededException`.
- `X-Neptune-Detail` – kosong untuk permintaan yang berhasil. Dalam kasus kesalahan, ia berisi pesan kesalahan JSON. Karena hanya karakter ASCII yang diperbolehkan dalam nilai header HTTP, string JSON di-enkode dengan URL.

> **Note**
>
> Neptune saat ini tidak mendukung kompresi `gzip` respons bongkahan. Jika klien meminta kedua encoding bongkahan dan kompresi pada saat yang sama, Neptune melewatkan kompresi.

Klien Gremlin berbasis Java untuk digunakan dengan Amazon Neptunus

Anda dapat menggunakan salah satu dari dua klien Gremlin berbasis Java open-source dengan Amazon Neptunus: klien Apache TinkerPop Java Gremlin, atau klien Gremlin untuk Amazon Neptunus.

Apache TinkerPop Java Gremlin klien

Apache TinkerPop Java gremlin-driver adalah standar, klien Gremlin resmi yang bekerja dengan database grafik -enabled. TinkerPop Gunakan klien ini ketika Anda membutuhkan kompatibilitas maksimum dengan ruang TinkerPop pengembangan yang lebih luas, saat Anda bekerja dengan beberapa sistem database grafik, atau ketika Anda tidak memerlukan manajemen cluster lanjutan dan fitur load balancing khusus untuk Neptunus. Klien ini juga cocok untuk aplikasi sederhana yang terhubung ke satu instance Neptunus atau ketika Anda lebih suka menangani load balancing di tingkat infrastruktur daripada di dalam klien.

> **Important**
>
> Memilih versi driver Apache TinkerPop Gremlin yang benar sangat penting untuk kompatibilitas dengan versi mesin Neptunus Anda. Menggunakan versi yang tidak kompatibel dapat mengakibatkan kegagalan koneksi atau perilaku yang tidak terduga. Untuk informasi kompatibilitas versi terperinci, lihatMengakses grafik Neptune dengan Gremlin.

> **Note**
>
> Tabel yang membantu Anda menentukan TinkerPop versi Apache yang benar untuk digunakan dengan Neptunus telah dipindahkan ke. the section called “Gremlin” Tabel ini sebelumnya terletak di halaman ini selama bertahun-tahun dan sekarang lebih terpusat untuk referensi untuk semua bahasa pemrograman yang TinkerPop mendukung.

Klien Gremlin Java untuk Amazon Neptunus

Klien Gremlin untuk Amazon Neptunus adalah klien Gremlin berbasis Java open-source yang bertindak sebagai pengganti drop-in untuk klien Java standar. TinkerPop

Klien Neptunus Gremlin dioptimalkan untuk cluster Neptunus. Ini memungkinkan Anda mengelola distribusi lalu lintas di beberapa instance dalam sebuah cluster, dan beradaptasi dengan perubahan topologi cluster saat Anda menambahkan atau menghapus replika. Anda bahkan dapat mengonfigurasi klien untuk mendistribusikan permintaan di seluruh subset instance di klaster Anda, berdasarkan peran, tipe instans, zona ketersediaan (AZ), atau tag yang terkait dengan instance.

Versi terbaru dari klien Neptunus Gremlin Java tersedia di Maven Central.

Untuk informasi lebih lanjut tentang klien Neptunus Gremlin Java, lihat posting blog ini. Untuk contoh kode dan demo, lihat GitHub proyek klien.

Saat memilih versi klien Neptunus Gremlin, Anda perlu mempertimbangkan versi yang TinkerPop mendasarinya sehubungan dengan versi mesin Neptunus Anda. Lihat tabel kompatibilitas di the section called “Gremlin” untuk menentukan TinkerPop versi yang benar untuk mesin Neptunus Anda, lalu gunakan tabel berikut untuk memilih versi klien Neptunus Gremlin yang sesuai:

Kompatibilitas versi klien Neptunus Gremlin

Versi klien Neptunus Gremlin	TinkerPop versi
3.x	3.7.x (AWS SDK for Java 2.x/1.x)
2.1.x	3.7.x (AWS SDK for Java 1.x)
2.0.x	3.6.x
1.12	3.5.x

Menggunakan klien Java untuk terhubung ke instance DB Neptunus

Bagian berikut memandu Anda melalui menjalankan sampel Java lengkap yang terhubung ke instance DB Neptunus dan melakukan traversal Gremlin menggunakan klien Apache Gremlin. TinkerPop

Anda harus mengikuti petunjuk ini dari instans Amazon EC2 di virtual private cloud (VPC) yang sama seperti instans DB Neptune Anda.

Untuk menyambung ke Neptune menggunakan Java

1. Install Apache Maven pada instans EC2 Anda. Jika menggunakan Amazon Linux 2023 (lebih disukai), gunakan:

   ```
   sudo dnf update -y
   sudo dnf install maven -y
   ```

   Jika menggunakan Amazon Linux 2, unduh biner terbaru dari https://maven.apache.org/download.cgi:

   ```
   sudo yum remove maven -y
   wget https://dlcdn.apache.org/maven/maven-3/ <version>/binaries/apache-maven-
   <version>-bin.tar.gz
   sudo tar -xzf apache-maven-<version>-bin.tar.gz -C /opt/
   sudo ln -sf /opt/apache-maven-<version> /opt/maven
   echo 'export MAVEN_HOME=/opt/maven' >> ~/.bashrc
   echo 'export PATH=$MAVEN_HOME/bin:$PATH' >> ~/.bashrc
   source ~/.bashrc
   ```

2. Instal Java. Perpustakaan Gremlin membutuhkan Java 8 atau 11. Anda dapat menginstal Java 11 sebagai berikut:

   - Jika Anda menggunakan Amazon Linux 2 (AL2):

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

   - Jika Anda menggunakan Amazon Linux 2023 (AL2023):

     ```
     sudo yum install java-11-amazon-corretto-devel
     ```

   - Untuk distribusi lain, gunakan salah satu dari berikut ini yang sesuai:

     ```
     sudo yum install java-11-openjdk-devel
     ```

     atau:

     ```
     sudo apt-get install openjdk-11-jdk
     ```

3. Setel Java 11 sebagai runtime default pada instans EC2 Anda: Masukkan yang berikut ini untuk menetapkan Java 8 sebagai runtime default pada instans EC2 Anda:

```
sudo /usr/sbin/alternatives --config java
```

Saat diminta, masukkan nomor untuk Java 11.

4. Buat direktori baru bernama**`gremlinjava`**:

```
mkdir gremlinjava
cd gremlinjava
```

5. Di direktori `gremlinjava`, buat file `pom.xml`, lalu buka file tersebut dalam editor teks:

```
nano pom.xml
```

6. Salin JSON berikut ke dalam file `pom.xml` dan simpan filenya:

```
<project xmlns="https://maven.apache.org/POM/4.0.0"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://
maven.apache.org/maven-v4_0_0.xsd">
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.amazonaws</groupId>
  <artifactId>GremlinExample</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>GremlinExample</name>
  <url>https://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>org.apache.tinkerpop</groupId>
      <artifactId>gremlin-driver</artifactId>
      <version>3.7.2</version>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-jdk14</artifactId>
      <version>1.7.25</version>
    </dependency>
  </dependencies>
  <build>
```

```
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.5.1</version>
        <configuration>
          <source>11</source>
          <target>11</target>
        </configuration>
      </plugin>
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>exec-maven-plugin</artifactId>
          <version>1.3</version>
          <configuration>
            <executable>java</executable>
            <arguments>
              <argument>-classpath</argument>
              <classpath/>
              <argument>com.amazonaws.App</argument>
            </arguments>
            <mainClass>com.amazonaws.App</mainClass>
            <complianceLevel>1.11</complianceLevel>
            <killAfter>-1</killAfter>
          </configuration>
        </plugin>
    </plugins>
  </build>
</project>
```

> **Note**
>
> Jika Anda memodifikasi proyek Maven yang ada, dependensi yang diperlukan disorot dalam kode sebelumnya.

7. Buat subdirektori untuk kode sumber contoh (`src/main/java/com/amazonaws/`) dengan mengetik teks berikut pada baris perintah:

```
mkdir -p src/main/java/com/amazonaws/
```

8. Di direktori `src/main/java/com/amazonaws/`, buat file bernama `App.java`, lalu buka file tersebut dalam editor teks.

```
nano src/main/java/com/amazonaws/App.java
```

9. Salin hal berikut ke dalam file `App.java`. Ganti *`your-neptune-endpoint`* dengan alamat instans DB Neptunus Anda. Jangan sertakan prefiks `https://` di metode `addContactPoint`.

> **Note**
>
> Untuk informasi tentang menemukan nama host instans DB Neptune Anda, lihat Menghubungkan ke Titik Akhir Amazon Neptune..

```
package com.amazonaws;
import org.apache.tinkerpop.gremlin.driver.Cluster;
import
 org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversalSource;
import org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal;
import static
 org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource.traversal;
import org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteConnection;
import org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.__;
import org.apache.tinkerpop.gremlin.structure.T;

public class App
{
  public static void main( String[] args )
  {
    Cluster.Builder builder = Cluster.build();
    builder.addContactPoint("your-neptune-endpoint");
    builder.port(8182);
    builder.enableSsl(true);

    Cluster cluster = builder.create();

    GraphTraversalSource g =
 traversal().withRemote(DriverRemoteConnection.using(cluster));

    // Add a vertex.
    // Note that a Gremlin terminal step, e.g. iterate(), is required to make a
 request to the remote server.
    // The full list of Gremlin terminal steps is at https://tinkerpop.apache.org/
docs/current/reference/#terminal-steps
```

```
    g.addV("Person").property("Name", "Justin").iterate();

    // Add a vertex with a user-supplied ID.
    g.addV("Custom Label").property(T.id, "CustomId1").property("name", "Custom id
 vertex 1").iterate();
    g.addV("Custom Label").property(T.id, "CustomId2").property("name", "Custom id
 vertex 2").iterate();

    g.addE("Edge Label").from(__.V("CustomId1")).to(__.V("CustomId2")).iterate();

    // This gets the vertices, only.
    GraphTraversal t = g.V().limit(3).elementMap();

    t.forEachRemaining(
      e ->  System.out.println(t.toList())
    );

    cluster.close();
  }
}
```

Untuk bantuan menghubungkan ke Neptunus SSL/TLS dengan (yang diperlukan), lihat. Konfigurasi SSL/TLS

10. Kompilasikan dan jalankan sampel menggunakan perintah Maven berikut:

```
mvn compile exec:exec
```

Contoh sebelumnya mengembalikan peta kunci dan nilai-nilai masing-masing properti untuk dua vertex pertama dalam grafik menggunakan traversal `g.V().limit(3).elementMap()`. Untuk mengajukan kueri untuk sesuatu yang lain, ganti dengan traversal Gremlin lain dengan salah satu metode ending yang tepat.

> **Note**
>
> Bagian akhir dari kueri Gremlin, `.toList()`, diperlukan untuk mengirimkan traversal ke server untuk evaluasi. Jika Anda tidak menyertakan metode tersebut atau metode setara lain, kueri tidak diserahkan ke instans DB Neptune.
> Anda juga harus menambahkan ending yang tepat ketika Anda menambahkan sebuah vertex atau edge, seperti ketika Anda menggunakan langkah `addV( )`.

Metode berikut mengirimkan kueri ke instans DB Neptune:

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

Konfigurasi SSL/TLS untuk klien Gremlin Java

Neptunus harus diaktifkan SSL/TLS secara default. Biasanya, jika driver Java dikonfigurasi dengan`enableSsl(true)`, ia dapat terhubung ke Neptunus tanpa harus mengatur `keyStore()` atau `trustStore()` dengan salinan lokal sertifikat.

Namun, jika instans yang Anda sambungkan tidak memiliki koneksi internet untuk memverifikasi sertifikat publik, atau jika sertifikat yang Anda gunakan tidak bersifat publik, Anda dapat mengambil langkah-langkah berikut untuk mengonfigurasi salinan sertifikat lokal:

Menyiapkan salinan sertifikat lokal untuk mengaktifkan SSL/TLS

1. Unduh dan instal keytool dari Oracle. Ini akan membuat pengaturan toko kunci lokal jauh lebih mudah.
2. Unduh sertifikat `SFSRootCAG2.pem` CA (Gremlin Java SDK memerlukan sertifikatuntuk memverifikasi sertifikat jarak jauh):

   ```
   wget https://www.amazontrust.com/repository/SFSRootCAG2.pem
   ```

3. Buat toko kunci dalam JKS atau PKCS12 format. Contoh ini menggunakan JKS. Jawab pertanyaan-pertanyaan yang mengikuti pada prompt. Kata sandi yang Anda buat di sini akan dibutuhkan nanti:

   ```
   keytool -genkey -alias (host name) -keyalg RSA -keystore server.jks
   ```

4. Impor `SFSRootCAG2.pem` file yang Anda unduh ke toko kunci yang baru dibuat:

   ```
   keytool -import -keystore server.jks -file .pem
   ```

5. Konfigurasikan `Cluster` objek secara terprogram:

```
Cluster cluster = Cluster.build("(your neptune endpoint)")
                         .port(8182)
                         .enableSSL(true)
                         .keyStore('server.jks')
                         .keyStorePassword("(the password from step 2)")
                         .create();
```

Anda dapat melakukan hal yang sama dalam file konfigurasi jika Anda mau, seperti yang mungkin Anda lakukan dengan konsol Gremlin:

```
hosts: [(your neptune endpoint)]
port: 8182
connectionPool: { enableSsl: true, keyStore: server.jks, keyStorePassword: (the
 password from step 2) }
serializer: { className:
 org.apache.tinkerpop.gremlin.util.ser.GraphBinaryMessageSerializerV1, config:
 { serializeResultToString: true }}
```

Autentikasi IAM

Neptunus mendukung otentikasi IAM untuk mengontrol akses ke cluster DB Anda. Jika autentikasi IAM diaktifkan, Anda harus menggunakan penandatanganan Signature Version 4 untuk mengautentikasi permintaan Anda. Untuk petunjuk rinci dan contoh kode untuk menghubungkan dari klien Java, lihatMenghubungkan ke database Amazon Neptunus menggunakan IAM dengan Gremlin Java.

Contoh Java tentang menghubungkan ke instans DB Neptune dengan logika sambungan ulang

Contoh Java berikut menunjukkan cara terhubung ke klien Gremlin dengan logika sambungan ulang untuk pulih dari pemutusan sambungan tak terduga.

Contoh ini memiliki dependensi berikut:

```
<dependency>
    <groupId>org.apache.tinkerpop</groupId>
    <artifactId>gremlin-driver</artifactId>
    <version>${gremlin.version}</version>
```

```
</dependency>

<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>amazon-neptune-sigv4-signer</artifactId>
    <version>${sig4.signer.version}</version>
</dependency>

<dependency>
    <groupId>com.evanlennick</groupId>
    <artifactId>retry4j</artifactId>
    <version>0.15.0</version>
</dependency>
```

Berikut adalah kode sampelnya:

> **Important**
>
> `CallExecutor`Dari Retry4j mungkin tidak aman untuk utas. Pertimbangkan agar setiap utas menggunakan `CallExecutor` instancenya sendiri, atau gunakan pustaka percobaan ulang yang berbeda.

> **Note**
>
> Contoh berikut telah diperbarui untuk menyertakan penggunaan requestInterceptor (). Ini ditambahkan di TinkerPop 3.6.6. Sebelum TinkerPop versi 3.6.6, contoh kode menggunakan HandshakeInterceptor (), yang tidak digunakan lagi dengan rilis itu.

```
public static void main(String args[]) {
  boolean useIam = true;

  // Create Gremlin cluster and traversal source
  Cluster.Builder builder = Cluster.build()
         .addContactPoint(System.getenv("neptuneEndpoint"))
         .port(Integer.parseInt(System.getenv("neptunePort")))
         .enableSsl(true)
         .minConnectionPoolSize(1)
         .maxConnectionPoolSize(1)
         .serializer(Serializers.GRAPHBINARY_V1D0)
```

```
        .reconnectInterval(2000);

 if (useIam) {
     builder.requestInterceptor( r -> {
        try {
           NeptuneNettyHttpSigV4Signer sigV4Signer =
                       new NeptuneNettyHttpSigV4Signer("(your region)", new
DefaultAWSCredentialsProviderChain());
           sigV4Signer.signRequest(r);
        } catch (NeptuneSigV4SignerException e) {
           throw new RuntimeException("Exception occurred while signing the request",
e);
        }
        return r;
     });
  }

 Cluster cluster = builder.create();

 GraphTraversalSource g = AnonymousTraversalSource
     .traversal()
     .withRemote(DriverRemoteConnection.using(cluster));

 // Configure retries
 RetryConfig retryConfig = new RetryConfigBuilder()
     .retryOnCustomExceptionLogic(getRetryLogic())
     .withDelayBetweenTries(1000, ChronoUnit.MILLIS)
     .withMaxNumberOfTries(5)
     .withFixedBackoff()
     .build();

 @SuppressWarnings("unchecked")
 CallExecutor<Object> retryExecutor = new CallExecutorBuilder<Object>()
     .config(retryConfig)
     .build();

 // Do lots of queries
 for (int i = 0; i < 100; i++){
   String id = String.valueOf(i);

   @SuppressWarnings("unchecked")
   Callable<Object> query = () -> g.V(id)
       .fold()
       .coalesce(
```

```
            unfold(),
            addV("Person").property(T.id, id))
        .id().next();

    // Retry query
    // If there are connection failures, the Java Gremlin client will automatically
    // attempt to reconnect in the background, so all we have to do is wait and retry.
    Status<Object> status = retryExecutor.execute(query);

    System.out.println(status.getResult().toString());
  }

  cluster.close();
}

private static Function<Exception, Boolean> getRetryLogic() {

  return e -> {

    Class<? extends Exception> exceptionClass = e.getClass();

    StringWriter stringWriter = new StringWriter();
    String message = stringWriter.toString();


    if (RemoteConnectionException.class.isAssignableFrom(exceptionClass)){
      System.out.println("Retrying because RemoteConnectionException");
      return true;
    }

    // Check for connection issues
    if (message.contains("Timed out while waiting for an available host") ||
        message.contains("Timed-out") && message.contains("waiting for connection on
 Host") ||
        message.contains("Connection to server is no longer active") ||
        message.contains("Connection reset by peer") ||
        message.contains("SSLEngine closed already") ||
        message.contains("Pool is shutdown") ||
        message.contains("ExtendedClosedChannelException") ||
        message.contains("Broken pipe") ||
        message.contains(System.getenv("neptuneEndpoint")))
    {
      System.out.println("Retrying because connection issue");
      return true;
```

```
    };

    // Concurrent writes can sometimes trigger a ConcurrentModificationException.
    // In these circumstances you may want to backoff and retry.
    if (message.contains("ConcurrentModificationException")) {
      System.out.println("Retrying because ConcurrentModificationException");
      return true;
    }

    // If the primary fails over to a new instance, existing connections to the old
 primary will
    // throw a ReadOnlyViolationException. You may want to back and retry.
    if (message.contains("ReadOnlyViolationException")) {
      System.out.println("Retrying because ReadOnlyViolationException");
      return true;
    }

    System.out.println("Not a retriable error");
    return false;
  };
}
```

Menggunakan Python untuk terhubung ke instans DB Neptune

> **Important**
>
> Memilih versi driver Apache TinkerPop Gremlin yang benar sangat penting untuk kompatibilitas dengan versi mesin Neptunus Anda. Menggunakan versi yang tidak kompatibel dapat mengakibatkan kegagalan koneksi atau perilaku yang tidak terduga. Untuk informasi kompatibilitas versi terperinci, lihatMengakses grafik Neptune dengan Gremlin.

Bagian berikut memandu Anda melalui proses berjalannya sampel Python yang menyambungkan ke instans Amazon Neptune DB dan melakukan traversal Gremlin.

Anda harus mengikuti petunjuk ini dari instans Amazon EC2 di virtual private cloud (VPC) yang sama seperti instans DB Neptune Anda.

Sebelum memulai, lakukan hal berikut:

- Unduh dan pasang Python 3.6 atau yang lebih baru dari Situs web Python.org.

- Pastikan Anda memiliki pip diinstal. Jika Anda tidak punya pip atau Anda tidak yakin, lihat Apakah saya perlu menginstal pip? di dokumentasi pip.
- Jika instalasi Python Anda belum memilikinya, unduh `futures` seperti berikut: `pip install futures`

Untuk menyambung ke Neptune menggunakan Python

1. Masukkan hal berikut untuk menginstal paket `gremlinpython`:

```
pip install --user gremlinpython
```

2. Buat file bernama `gremlinexample.py`, lalu buka file tersebut dalam editor teks.
3. Salin hal berikut ke dalam file `gremlinexample.py`. Ganti *`your-neptune-endpoint`* dengan alamat cluster DB Neptunus Anda *`your-neptune-port`* dan dengan port cluster DB Neptunus Anda (default:8182).

   Untuk informasi tentang menemukan alamat instans DB Neptune Anda, lihat bagian Menghubungkan ke Titik Akhir Amazon Neptune..

   Contoh di bawah ini menunjukkan bagaimana untuk terhubung dengan Gremlin Python.

```
import boto3
import os
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection
from gremlin_python.process.anonymous_traversal import traversal

database_url = "wss://your-neptune-endpoint:your-neptune-port/gremlin"

remoteConn = DriverRemoteConnection(database_url, "g")

g = traversal().withRemote(remoteConn)

print(g.inject(1).toList())
remoteConn.close()
```

4. Masukkan perintah berikut untuk menjalankan sampel:

```
python gremlinexample.py
```

Kueri Gremlin pada akhir contoh ini mengembalikan vertex (`g.V().limit(2)`) dalam daftar. Daftar ini kemudian dicetak dengan fungsi `print` standar Python.

> **Note**
>
> Bagian akhir dari kueri Gremlin, `toList()`, diperlukan untuk mengirimkan traversal ke server untuk evaluasi. Jika Anda tidak menyertakan metode tersebut atau metode setara lain, kueri tidak diserahkan ke instans DB Neptune.

Metode berikut mengirimkan kueri ke instans DB Neptune:

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

Contoh sebelumnya mengembalikan dua vertex pertama dalam grafik menggunakan traversal `g.V().limit(2).toList()`. Untuk mengajukan kueri untuk sesuatu yang lain, ganti dengan traversal Gremlin lain dengan salah satu metode ending yang tepat.

Autentikasi IAM

Neptunus mendukung otentikasi IAM untuk mengontrol akses ke cluster DB Anda. Jika autentikasi IAM diaktifkan, Anda harus menggunakan penandatanganan Signature Version 4 untuk mengautentikasi permintaan Anda. Untuk petunjuk rinci dan contoh kode untuk menghubungkan dari klien Python, lihat. Menghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Python Gremlin

Gunakan .NET untuk terhubung ke instans DB Neptune

> **Important**
>
> Memilih versi driver Apache TinkerPop Gremlin yang benar sangat penting untuk kompatibilitas dengan versi mesin Neptunus Anda. Menggunakan versi yang tidak kompatibel dapat mengakibatkan kegagalan koneksi atau perilaku yang tidak terduga. Untuk informasi kompatibilitas versi terperinci, lihatMengakses grafik Neptune dengan Gremlin.

Bagian berikut berisi contoh kode yang ditulis dalam C# yang menyambung ke instans DB Neptune dan melakukan traversal Gremlin.

Sambungan ke Amazon Neptune harus berasal dari Instans Amazon EC2 di virtual private cloud (VPC) yang sama seperti instans DB Neptune Anda. Kode sampel ini diuji pada instans Amazon EC2 yang menjalankan Ubuntu.

Sebelum memulai, lakukan hal berikut:

- Instal .NET pada instans Amazon EC2. Untuk mendapatkan petunjuk cara menginstal .NET di beberapa sistem operasi, termasuk Windows, Linux, dan macOS, lihat Memulai dengan .NET.
- Instal Gremlin.NET dengan menjalankan `dotnet add package gremlin.net` untuk paket Anda. Untuk informasi selengkapnya, lihat Gremlin.net di dokumentasi. TinkerPop

Untuk menyambung ke Neptune menggunakan Gremlin.NET

1. Buat proyek .NET baru.

   ```
   dotnet new console -o gremlinExample
   ```

2. Ubah direktori ke direktori proyek baru.

   ```
   cd gremlinExample
   ```

3. Salin hal berikut ke dalam file `Program.cs`. Ganti *`your-neptune-endpoint`* dengan alamat instans DB Neptunus Anda.

Untuk informasi tentang menemukan alamat instans DB Neptune Anda, lihat bagian Menghubungkan ke Titik Akhir Amazon Neptune..

```
using System;
using System.Threading.Tasks;
using System.Collections.Generic;
using Gremlin.Net;
using Gremlin.Net.Driver;
using Gremlin.Net.Driver.Remote;
using Gremlin.Net.Structure;
using static Gremlin.Net.Process.Traversal.AnonymousTraversalSource;
namespace gremlinExample
{
  class Program
  {
    static void Main(string[] args)
    {
      try
      {
        var endpoint = "your-neptune-endpoint";
        // This uses the default Neptune and Gremlin port, 8182
        var gremlinServer = new GremlinServer(endpoint, 8182, enableSsl: true );
        var gremlinClient = new GremlinClient(gremlinServer);
        var remoteConnection = new DriverRemoteConnection(gremlinClient, "g");
        var g = Traversal().WithRemote(remoteConnection);
        g.AddV("Person").Property("Name", "Justin").Iterate();
        g.AddV("Custom Label").Property("name", "Custom id vertex 1").Iterate();
        g.AddV("Custom Label").Property("name", "Custom id vertex 2").Iterate();
        var output = g.V().Limit<Vertex>(3).ToList();
        foreach(var item in output) {
            Console.WriteLine(item);
        }
      }
      catch (Exception e)
      {
          Console.WriteLine("{0}", e);
      }
    }
  }
}
```

4. Masukkan perintah berikut untuk menjalankan sampel:

```
dotnet run
```

Kueri Gremlin pada akhir contoh ini mengembalikan jumlah vertex tunggal untuk tujuan pengujian. Ini kemudian dicetak ke konsol.

> **Note**
>
> Bagian akhir dari kueri Gremlin, `Next()`, diperlukan untuk mengirimkan traversal ke server untuk evaluasi. Jika Anda tidak menyertakan metode tersebut atau metode setara lain, kueri tidak diserahkan ke instans DB Neptune.

Metode berikut mengirimkan kueri ke instans DB Neptune:

- `ToList()`
- `ToSet()`
- `Next()`
- `NextTraverser()`
- `Iterate()`

Gunakan `Next()` jika Anda membutuhkan hasil kueri agar diserialkan dan dikembalikan, atau `Iterate()` jika tidak.

Contoh sebelumnya mengembalikan daftar dengan menggunakan traversal. `g.V().Limit(3).ToList()` Untuk mengajukan kueri untuk sesuatu yang lain, ganti dengan traversal Gremlin lain dengan salah satu metode ending yang tepat.

Autentikasi IAM

Neptunus mendukung otentikasi IAM untuk mengontrol akses ke cluster DB Anda. Jika autentikasi IAM diaktifkan, Anda harus menggunakan penandatanganan Signature Version 4 untuk mengautentikasi permintaan Anda. Untuk petunjuk rinci dan contoh kode untuk menghubungkan dari klien.NET, lihatMenghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Gremlin.NET.

Gunakan Node.js untuk terhubung ke instans DB Neptune

> **Important**
>
> Memilih versi driver Apache TinkerPop Gremlin yang benar sangat penting untuk kompatibilitas dengan versi mesin Neptunus Anda. Menggunakan versi yang tidak kompatibel dapat mengakibatkan kegagalan koneksi atau perilaku yang tidak terduga. Untuk informasi kompatibilitas versi terperinci, lihatMengakses grafik Neptune dengan Gremlin.

Bagian berikut memandu Anda melalui proses berjalannya sampel Node.js yang menyambungkan ke instans Amazon Neptune DB dan melakukan traversal Gremlin.

Anda harus mengikuti petunjuk ini dari instans Amazon EC2 di virtual private cloud (VPC) yang sama seperti instans DB Neptune Anda.

Sebelum memulai, lakukan hal berikut:

- Pastikan Node.js versi 8.11 atau lebih tinggi yang diinstal. Jika bukan, unduh dan instal Node.js dari Situs web Nodejs.org.

Untuk menyambung ke Neptune menggunakan Node.js

1. Masukkan hal berikut untuk menginstal paket `gremlin-javascript`:

   ```
   npm install gremlin
   ```

2. Buat file bernama `gremlinexample.js`, lalu buka file tersebut dalam editor teks.
3. Salin hal berikut ke dalam file `gremlinexample.js`. Ganti *`your-neptune-endpoint`* dengan alamat instans DB Neptunus Anda.

   Untuk informasi tentang menemukan alamat instans DB Neptune Anda, lihat bagian Menghubungkan ke Titik Akhir Amazon Neptune..

   ```
   const gremlin = require('gremlin');
   const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;
   const Graph = gremlin.structure.Graph;

   dc = new DriverRemoteConnection('wss://your-neptune-endpoint:8182/gremlin',{});
   ```

```
const graph = new Graph();
const g = graph.traversal().withRemote(dc);

g.V().limit(1).count().next().
    then(data => {
        console.log(data);
        dc.close();
    }).catch(error => {
        console.log('ERROR', error);
        dc.close();
    });
```

4. Masukkan perintah berikut untuk menjalankan sampel:

```
node gremlinexample.js
```

Contoh sebelumnya mengembalikan jumlah vertex tunggal dalam grafik menggunakan traversal `g.V().limit(1).count().next()`. Untuk mengajukan kueri untuk sesuatu yang lain, ganti dengan traversal Gremlin lain dengan salah satu metode ending yang tepat.

> **Note**
>
> Bagian akhir dari kueri Gremlin, `next()`, diperlukan untuk mengirimkan traversal ke server untuk evaluasi. Jika Anda tidak menyertakan metode tersebut atau metode setara lain, kueri tidak diserahkan ke instans DB Neptune.

Metode berikut mengirimkan kueri ke instans DB Neptune:

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

Gunakan `next()` jika Anda membutuhkan hasil kueri agar diserialkan dan dikembalikan, atau `iterate()` jika tidak.

> **Important**
>
> Ini adalah contoh Node.js mandiri. Jika Anda berencana untuk menjalankan kode seperti ini dalam suatu AWS Lambda fungsi, lihat Contoh fungsi Lambda detail tentang penggunaan JavaScript secara efisien dalam fungsi Lambda Neptunus.

Autentikasi IAM

Neptunus mendukung otentikasi IAM untuk mengontrol akses ke cluster DB Anda. Jika autentikasi IAM diaktifkan, Anda harus menggunakan penandatanganan Signature Version 4 untuk mengautentikasi permintaan Anda. Untuk instruksi terperinci dan contoh kode untuk menghubungkan dari JavaScript klien, lihatMenghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Gremlin JavaScript.

Menggunakan Go untuk terhubung ke instans DB Neptunus

> **Important**
>
> Memilih versi driver Apache TinkerPop Gremlin yang benar sangat penting untuk kompatibilitas dengan versi mesin Neptunus Anda. Menggunakan versi yang tidak kompatibel dapat mengakibatkan kegagalan koneksi atau perilaku yang tidak terduga. Untuk informasi kompatibilitas versi terperinci, lihatMengakses grafik Neptune dengan Gremlin.

> **Note**
>
> Versi 3.5.x gremlingo kompatibel dengan versi 3.4.x selama Anda hanya menggunakan TinkerPop fitur 3.4.x dalam kueri Gremlin yang Anda tulis.

Bagian berikut memandu Anda melalui menjalankan sampel Go yang terhubung ke instans Amazon Neptunus DB dan melakukan traversal Gremlin.

Anda harus mengikuti petunjuk ini dari instans Amazon EC2 di virtual private cloud (VPC) yang sama seperti instans DB Neptune Anda.

Sebelum memulai, lakukan hal berikut:

- Unduh dan instal Go 1.17 atau yang lebih baru dari situs web go.dev.

Untuk terhubung ke Neptunus menggunakan Go

1. Mulai dari direktori kosong, inisialisasi modul Go baru:

```
go mod init example.com/gremlinExample
```

2. Tambahkan gremlin-go sebagai dependensi modul baru Anda:

```
go get github.com/apache/tinkerpop/gremlin-go/v3/driver
```

3. Buat file bernama `gremlinExample.go` dan kemudian buka di editor teks.
4. Salin yang berikut ini ke dalam `gremlinExample.go` file, ganti *(your neptune endpoint)* dengan alamat instans DB Neptunus Anda:

```
package main

import (
  "fmt"
  gremlingo "github.com/apache/tinkerpop/gremlin-go/v3/driver"
)

func main() {
  // Creating the connection to the server.
  driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection("wss://(your
 neptune endpoint):8182/gremlin",
    func(settings *gremlingo.DriverRemoteConnectionSettings) {
      settings.TraversalSource = "g"
    })
  if err != nil {
    fmt.Println(err)
    return
  }
  // Cleanup
  defer driverRemoteConnection.Close()

  // Creating graph traversal
  g := gremlingo.Traversal_().WithRemote(driverRemoteConnection)

  // Perform traversal
  results, err := g.V().Limit(2).ToList()
```

```
  if err != nil {
    fmt.Println(err)
    return
  }
  // Print results
  for _, r := range results {
    fmt.Println(r.GetString())
  }
}
```

> **Note**
>
> Format sertifikat Neptunus TLS saat ini tidak didukung di Go 1.18+ dengan macOS, dan mungkin memberikan kesalahan 509 saat mencoba memulai koneksi. Untuk pengujian lokal, ini dapat dilewati dengan menambahkan “crypto/tls” ke impor dan memodifikasi pengaturan sebagai berikut: `DriverRemoteConnection`
>
> ```
> // Creating the connection to the server.
> driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection("wss://
> your-neptune-endpoint:8182/gremlin",
>   func(settings *gremlingo.DriverRemoteConnectionSettings) {
>       settings.TraversalSource = "g"
>       settings.TlsConfig = &tls.Config{InsecureSkipVerify: true}
>   })
> ```

5. Masukkan perintah berikut untuk menjalankan sampel:

```
go run gremlinExample.go
```

Query Gremlin di akhir contoh ini mengembalikan simpul (`g.V().Limit(2)`) dalam irisan. Irisan ini kemudian diulang dan dicetak dengan `fmt.Println` fungsi standar.

> **Note**
>
> Bagian akhir dari kueri Gremlin, `ToList()`, diperlukan untuk mengirimkan traversal ke server untuk evaluasi. Jika Anda tidak menyertakan metode tersebut atau metode setara lain, kueri tidak diserahkan ke instans DB Neptune.

Metode berikut mengirimkan kueri ke instans DB Neptune:

- `ToList()`
- `ToSet()`
- `Next()`
- `GetResultSet()`
- `Iterate()`

Contoh sebelumnya mengembalikan dua vertex pertama dalam grafik menggunakan traversal `g.V().Limit(2).ToList()`. Untuk mengajukan kueri untuk sesuatu yang lain, ganti dengan traversal Gremlin lain dengan salah satu metode ending yang tepat.

Autentikasi IAM

Neptunus mendukung otentikasi IAM untuk mengontrol akses ke cluster DB Anda. Jika autentikasi IAM diaktifkan, Anda harus menggunakan penandatanganan Signature Version 4 untuk mengautentikasi permintaan Anda. Untuk petunjuk rinci dan contoh kode untuk menghubungkan dari klien Go, lihatMenghubungkan ke database Amazon Neptunus menggunakan otentikasi IAM dengan Gremlin Go.

Menggunakan AWS SDK untuk menjalankan kueri Gremlin

Dengan AWS SDK, Anda dapat menjalankan kueri Gremlin terhadap grafik Neptunus Anda menggunakan bahasa pemrograman pilihan Anda. Neptunus data API SDK (`neptunedata`nama layanan) menyediakan tindakan untuk mengirimkan kueri ExecuteGremlinQueryGremlin.

Anda harus menjalankan contoh ini dari instans Amazon EC2 di virtual private cloud (VPC) yang sama dengan cluster DB Neptunus Anda, atau dari lokasi yang memiliki konektivitas jaringan ke titik akhir cluster Anda.

Tautan langsung ke dokumentasi referensi API untuk `neptunedata` layanan di setiap bahasa SDK dapat ditemukan di bawah ini:

Bahasa pemrograman	referensi API neptunedata
C++	https://sdk.amazonaws.com/cpp/api/LATEST/aws-cpp-sdk-neptunedata/html/annotated.html

Bahasa pemrograman	referensi API neptunedata
Go	https://docs.aws.amazon.com/sdk-for-go/api/service/neptunedata/
Java	https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/services/neptunedata/package-summary.html
JavaScript	https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/Package/-aws-sdk-client-neptunedata/
Kotlin	https://sdk.amazonaws.com/kotlin/api/latest/neptunedata/index.html
.NET	https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Neptunedata/NNeptunedata.html
PHP	https://docs.aws.amazon.com/aws-sdk-php/v3/api/namespace-Aws.Neptunedata.html
Python	https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/neptunedata.html
Ruby	https://docs.aws.amazon.com/sdk-for-ruby/v3/api/Aws/Neptunedata.html
Karat	https://crates.io/crates/aws-sdk-neptunedata
CLI	https://docs.aws.amazon.com/cli/latest/reference/neptunedata/

Contoh SDK Gremlin AWS

Contoh berikut menunjukkan cara menyiapkan `neptunedata` klien, menjalankan kueri Gremlin, dan mencetak hasilnya. Ganti *`YOUR_NEPTUNE_HOST`* dan *`YOUR_NEPTUNE_PORT`* dengan titik akhir dan port cluster DB Neptunus Anda.

⚠ Konfigurasi batas waktu sisi klien dan coba lagi

Batas waktu klien SDK mengontrol berapa lama klien menunggu respons. Itu tidak mengontrol berapa lama kueri berjalan di server. Jika waktu klien habis sebelum server selesai, kueri dapat terus berjalan di Neptunus sementara klien tidak memiliki cara untuk mengambil hasilnya.

Sebaiknya atur batas waktu baca sisi klien ke `0` (tanpa batas waktu) atau ke nilai yang setidaknya beberapa detik lebih lama dari pengaturan neptune_query_timeout sisi server di cluster DB Neptunus Anda. Ini memungkinkan Neptunus mengontrol saat waktu kueri habis. Kami juga merekomendasikan pengaturan upaya coba ulang maksimum ke 1 (tidak ada percobaan ulang). Jika SDK mencoba ulang kueri yang masih berjalan di server, itu dapat menghasilkan operasi duplikat. Ini sangat penting untuk kueri mutasi, di mana percobaan lagi dapat menyebabkan duplikat penulisan yang tidak diinginkan.

Python

1. 4. Ikuti instruksi instalasi untuk menginstal Boto3.
2. Buat file bernama `gremlinExample.py` dan tempel kode berikut:

```
import boto3
import json
from botocore.config import Config

# Disable the client-side read timeout and retries so that
# Neptune's server-side neptune_query_timeout controls query duration.
client = boto3.client(
    'neptunedata',
    endpoint_url=f'https://YOUR_NEPTUNE_HOST:YOUR_NEPTUNE_PORT',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

# Use the untyped GraphSON v3 serializer for a cleaner JSON response.
response = client.execute_gremlin_query(
    gremlinQuery='g.V().limit(1)',
    serializer='application/vnd.gremlin-v3.0+json;types=false'
)

print(json.dumps(response['result'], indent=2))
```

3. Jalankan contoh: `python gremlinExample.py`

Java

1. Ikuti petunjuk penginstalan untuk menyiapkan AWS SDK for Java.
2. Gunakan kode berikut untuk menyiapkan`NeptunedataClient`, menjalankan kueri Gremlin, dan mencetak hasilnya:

```
import java.net.URI;
import java.time.Duration;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.services.neptunedata.NeptunedataClient;
import
 software.amazon.awssdk.services.neptunedata.model.ExecuteGremlinQueryRequest;
import
 software.amazon.awssdk.services.neptunedata.model.ExecuteGremlinQueryResponse;

// Disable the client-side timeout and retries so that
// Neptune's server-side neptune_query_timeout controls query duration.
NeptunedataClient client = NeptunedataClient.builder()
    .endpointOverride(URI.create("https://YOUR_NEPTUNE_HOST:YOUR_NEPTUNE_PORT"))
    .overrideConfiguration(ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ZERO)
        .retryPolicy(RetryPolicy.none())
        .build())
    .build();

// Use the untyped GraphSON v3 serializer for a cleaner JSON response.
ExecuteGremlinQueryRequest request = ExecuteGremlinQueryRequest.builder()
    .gremlinQuery("g.V().limit(1)")
    .serializer("application/vnd.gremlin-v3.0+json;types=false")
    .build();

ExecuteGremlinQueryResponse response = client.executeGremlinQuery(request);

System.out.println(response.result().toString());
```

JavaScript

1. Ikuti petunjuk penginstalan untuk menyiapkan AWS SDK. JavaScript Instal paket klien neptunedata:. `npm install @aws-sdk/client-neptunedata`
2. Buat file bernama `gremlinExample.js` dan tempel kode berikut:

```
import { NeptunedataClient, ExecuteGremlinQueryCommand } from "@aws-sdk/client-
neptunedata";
import { NodeHttpHandler } from "@smithy/node-http-handler";

const config = {
    endpoint: "https://YOUR_NEPTUNE_HOST:YOUR_NEPTUNE_PORT",
    // Disable the client-side request timeout so that
    // Neptune's server-side neptune_query_timeout controls query duration.
    requestHandler: new NodeHttpHandler({
        requestTimeout: 0
    }),
    maxAttempts: 1
};

const client = new NeptunedataClient(config);

// Use the untyped GraphSON v3 serializer for a cleaner JSON response.
const input = {
    gremlinQuery: "g.V().limit(1)",
    serializer: "application/vnd.gremlin-v3.0+json;types=false"
};

const command = new ExecuteGremlinQueryCommand(input);
const response = await client.send(command);

console.log(JSON.stringify(response, null, 2));
```

3. Jalankan contoh: `node gremlinExample.js`

Petunjuk kueri Gremlin

Gunakan petunjuk kueri untuk menentukan optimasi dan evaluasi strategi untuk kueri Gremlin tertentu di Amazon Neptune.

Petunjuk kueri ditentukan dengan menambahkan langkah `withSideEffect` untuk kueri dengan sintaks berikut.

```
g.withSideEffect(hint, value)
```

- petunjuk – Mengidentifikasi jenis petunjuk yang akan diterapkan.
- nilai – Menentukan perilaku aspek sistem yang sedang dipertimbangkan.

Sebagai contoh, hal berikut ini menunjukkan bagaimana cara memasukkan petunjuk `repeatMode`dalam traversal Gremlin.

> **Note**
>
> Semua efek samping petunjuk kueri Gremlin diprefiks dengan `Neptune#`.

```
g.withSideEffect('Neptune#repeatMode',
 'DFS').V("3").repeat(out()).times(10).limit(1).path()
```

Kueri sebelumnya menginstruksikan mesin Neptune untuk melakukan traversal pada grafik Depth First (`DFS`) ketimbang Neptune default, Breadth First (`BFS`).

Bagian berikut menyediakan informasi selengkapnya tentang petunjuk kueri yang tersedia dan penggunaannya.

Topik

- Petunjuk kueri repeatMode Gremlin
- Petunjuk kueri noReordering Gremlin
- Petunjuk permintaan TypePromotion Gremlin
- Petunjuk kueri useDFE Gremlin
- Petunjuk kueri Gremlin untuk menggunakan cache hasil

Petunjuk kueri repeatMode Gremlin

Petunjuk kueri `repeatMode` Neptune menentukan bagaimana mesin Neptune mengevaluasi langkah `repeat()` dalam traversal Gremlin: breadth first, depth first, atau chunked depth first.

Mode evaluasi langkah `repeat()` penting ketika digunakan untuk menemukan atau mengikuti jalur, bukan hanya mengulangi langkah dalam jumlah pengulangan terbatas.

Sintaksis

Petunjuk kueri `repeatMode` ditentukan dengan menambahkan langkah `withSideEffect` untuk kueri.

```
g.withSideEffect('Neptune#repeatMode', 'mode').gremlin-traversal
```

> **Note**
>
> Semua efek samping petunjuk kueri Gremlin diprefiks dengan `Neptune#`.

Mode yang Tersedia

- `BFS`

  Pencarian Breadth-First

  Mode eksekusi default untuk langkah `repeat()`. Ini mendapat semua node saudara sebelum pergi lebih dalam sepanjang jalur.

  Versi ini intensif memori dan perbatasan bisa menjadi sangat besar. Ada risiko yang lebih tinggi bahwa kueri akan kehabisan memori dan dibatalkan oleh mesin Neptune. Ini paling cocok dengan implementasi Gremlin lainnya.
- `DFS`

  Pencarian Depth-First

  Mengikuti setiap jalur ke kedalaman maksimum sebelum beralih ke solusi berikutnya.

  Ini menggunakan lebih sedikit memori. Ini dapat memberikan performa yang lebih baik dalam situasi seperti menemukan jalur tunggal dari titik awal ke beberapa hop.
- `CHUNKED_DFS`

  Pencarian Chunked Depth-First

Pendekatan hibrida yang mengeksplorasi depth-first grafik dalam potongan 1.000 node, bukan 1 node (DFS) atau semua node (BFS)).

Mesin Neptune akan mendapatkan hingga 1.000 node pada setiap tingkat sebelum mengikuti jalur yang lebih dalam.

Ini adalah pendekatan yang seimbang antara kecepatan dan penggunaan memori.

Hal ini juga berguna jika Anda ingin menggunakan BFS, tapi kueri menggunakan terlalu banyak memori.

Contoh

Bagian berikut menjelaskan efek dari mode pengulangan pada traversal Gremlin.

Di Neptune, mode default untuk langkah `repeat()` adalah untuk melakukan strategi eksekusi breadth-first (BFS) untuk semua traversal.

Dalam kebanyakan kasus, TinkerGraph implementasi menggunakan strategi eksekusi yang sama, tetapi dalam beberapa kasus mengubah eksekusi traversal.

Misalnya, TinkerGraph implementasi memodifikasi query berikut.

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

Langkah `repeat()` dalam traversal ini di-“unrolled” ke dalam traversal berikut, yang menghasilkan strategi depth-first (DFS).

```
g.V(<id>).out().out().out().out().out().out().out().out().out().out().limit(1).path()
```

> **Important**
>
> Mesin permintaan Neptune tidak melakukan ini secara otomatis.

Breadth-first (BFS) adalah strategi eksekusi default, dan mirip dengan TinkerGraph dalam kebanyakan kasus. Namun, ada kasus-kasus tertentu di mana strategi depth (DFS) lebih baik.

BFS (Default)

Breadth-first (BFS) adalah strategi eksekusi default untuk operator `repeat()`.

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

Mesin Neptune sepenuhnya mengeksplorasi perbatasan sembilan hop pertama sebelum menemukan sepuluh hop solusi. Hal ini efektif dalam banyak kasus, seperti kueri jalur terpendek.

Namun, untuk contoh sebelumnya, traversal akan jauh lebih cepat menggunakan mode depth-first (DFS) untuk operator `repeat()`.

DFS

Kueri berikut menggunakan mode depth-first (DFS) untuk operator `repeat()`.

```
g.withSideEffect("Neptune#repeatMode", "DFS").V("3").repeat(out()).times(10).limit(1)
```

Ini mengikuti setiap solusi individu ke kedalaman maksimum sebelum menjelajahi solusi berikutnya.

Petunjuk kueri noReordering Gremlin

Ketika Anda mengirimkan traversal Gremlin, mesin kueri Neptune menyelidiki struktur traversal dan mengurutkan ulang bagian dari kueri, mencoba untuk meminimalkan jumlah pekerjaan yang diperlukan untuk evaluasi dan waktu respons kueri. Sebagai contoh, sebuah traversal dengan beberapa kendala, seperti beberapa langkah `has()`, biasanya tidak dievaluasi dalam urutan yang diberikan. Sebaliknya traversal diurutkan ulang setelah kueri diperiksa dengan analisis statis.

Mesin kueri Neptune mencoba untuk mengidentifikasi kendala mana yang lebih selektif dan menjalankannya terlebih dulu. Hal ini sering menghasilkan performa yang lebih baik, tetapi urutan yang dipilih Neptune untuk mengevaluasi kueri mungkin tidak selalu optimal.

Jika Anda tahu karakteristik data yang tepat dan ingin secara manual mendikte urutan eksekusi kueri, Anda dapat menggunakan petunjuk kueri `noReordering` Neptune untuk menentukan bahwa traversal dievaluasi dalam urutan yang diberikan.

Sintaksis

Petunjuk kueri `noReordering` ditentukan dengan menambahkan langkah `withSideEffect` untuk kueri.

```
g.withSideEffect('Neptune#noReordering', true or false).gremlin-traversal
```

> **Note**
>
> Semua efek samping petunjuk kueri Gremlin diprefiks dengan `Neptune#`.

Nilai yang Tersedia

- `true`
- `false`

Petunjuk permintaan TypePromotion Gremlin

Ketika Anda mengirimkan traversal Gremlin yang memfilter di atas nilai atau jangkauan numerik, mesin kueri Neptune biasanya harus menggunakan promosi jenis ketika mengeksekusi kueri. Ini berarti bahwa mesin harus memeriksa nilai-nilai dari setiap jenis yang dapat memegang nilai yang Anda gunakan untuk menyaring.

Misalnya, jika Anda menyaring untuk nilai yang sama dengan 55, mesin harus mencari bilangan bulat yang sama dengan 55, bilangan bulat panjang sama dengan 55L, mengapung sama dengan 55.0, dan sebagainya. Setiap promosi jenis memerlukan pencarian tambahan pada penyimpanan, yang dapat menyebabkan permintaan yang tampaknya sederhana tiba-tiba mengambil waktu yang lama untuk menyelesaikan.

Katakanlah Anda sedang mencari semua vertex dengan properti customer-age lebih besar dari 5:

```
g.V().has('customerAge', gt(5))
```

Untuk mengeksekusi traversal yang secara menyeluruh, Neptune harus memperluas kueri untuk memeriksa setiap jenis numerik yang dapat mempromosikan nilai Anda kueri. Dalam hal ini, filter `gt` harus diterapkan untuk setiap integer lebih dari 5, setiap panjang lebih dari 5L, setiap float lebih dari 5.0, dan setiap ganda lebih 5.0. Karena masing-masing promosi jenis ini memerlukan pencarian tambahan pada penyimpanan, Anda akan melihat beberapa filter per filter numerik ketika Anda menjalankan API `profile` Gremlin untuk kueri ini, dan akan memakan waktu lebih lama untuk menyelesaikan dari yang mungkin Anda harapkan.

Sering kali promosi jenis tidak perlu karena Anda tahu sebelumnya bahwa Anda hanya perlu menemukan nilai dari satu jenis tertentu. Ketika hal ini terjadi, Anda dapat mempercepat kueri Anda secara dramatis dengan menggunakan petunjuk kueri `typePromotion` untuk mematikan promosi jenis.

Sintaksis

Petunjuk kueri `typePromotion` ditentukan dengan menambahkan langkah `withSideEffect` untuk kueri.

```
g.withSideEffect('Neptune#typePromotion', true or false).gremlin-traversal
```

> **Note**
>
> Semua efek samping petunjuk kueri Gremlin diprefiks dengan `Neptune#`.

Nilai yang Tersedia

- `true`
- `false`

Untuk menonaktifkan promosi jenis untuk kueri di atas, Anda akan menggunakan:

```
g.withSideEffect('Neptune#typePromotion', false).V().has('customerAge', gt(5))
```

Petunjuk kueri useDFE Gremlin

Gunakan petunjuk kueri ini untuk mengaktifkan penggunaan DFE untuk mengeksekusi kueri. Secara default Neptunus tidak menggunakan DFE tanpa petunjuk kueri ini disetel ke, karena parameter instance neptune_dfe_query_engine default `true` ke. `viaQueryHint` Jika Anda menyetel parameter instance itu`enabled`, mesin DFE digunakan untuk semua kueri kecuali yang memiliki petunjuk `useDFE` kueri yang disetel ke. `false`

Contoh mengaktifkan DFE untuk kueri:

```
g.withSideEffect('Neptune#useDFE', true).V().out()
```

Petunjuk kueri Gremlin untuk menggunakan cache hasil

Petunjuk kueri berikut dapat digunakan saat cache hasil kueri diaktifkan.

Petunjuk kueri Gremlin **enableResultCache**

Petunjuk `enableResultCache` kueri dengan nilai `true` menyebabkan hasil kueri dikembalikan dari cache jika sudah di-cache. Jika tidak, ia mengembalikan hasil baru dan men-cache mereka sampai waktu mereka dihapus dari cache. Contoh:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Kemudian, Anda dapat mengakses hasil cache dengan mengeluarkan kueri yang persis sama lagi.

Jika nilai petunjuk kueri ini`false`, atau jika tidak ada, hasil kueri tidak di-cache. Namun, menyetelnya ke `false` tidak menghapus hasil cache yang ada. Untuk menghapus hasil cache, gunakan `invalidateResultCachekey` petunjuk `invalidateResultCache` atau.

Petunjuk kueri Gremlin **enableResultCacheWithTTL**

Petunjuk `enableResultCacheWithTTL` kueri juga mengembalikan hasil cache jika ada, tanpa mempengaruhi TTL hasil yang sudah ada di cache. Jika saat ini tidak ada hasil cache, kueri mengembalikan hasil baru dan menyimpannya dalam cache untuk waktu hidup (TTL) yang ditentukan oleh petunjuk `enableResultCacheWithTTL` kueri. Waktu untuk hidup ditentukan dalam hitungan detik. Misalnya, kueri berikut menentukan waktu untuk hidup enam puluh detik:

```
g.with('Neptune#enableResultCacheWithTTL', 60)
 .V().has('genre','drama').in('likes')
```

Sebelum 60 detik time-to-live selesai, Anda dapat menggunakan kueri yang sama (di sini,`g.V().has('genre','drama').in('likes')`) dengan petunjuk `enableResultCache` atau `enableResultCacheWithTTL` kueri untuk mengakses hasil yang di-cache.

> **Note**
>
> Waktu untuk hidup ditentukan dengan `enableResultCacheWithTTL` tidak mempengaruhi hasil yang telah di-cache.

- Jika hasil sebelumnya di-cache menggunakan`enableResultCache`, cache harus terlebih dahulu dihapus secara eksplisit sebelum `enableResultCacheWithTTL` menghasilkan hasil baru dan menyimpannya untuk TTL yang ditentukannya.
- Jika hasil sebelumnya di-cache menggunakan`enableResultCachewithTTL`, TTL sebelumnya harus kedaluwarsa terlebih dahulu sebelum `enableResultCacheWithTTL` menghasilkan hasil baru dan menyimpannya untuk TTL yang ditentukannya.

Setelah waktu untuk hidup berlalu, hasil cache untuk kueri dihapus, dan contoh berikutnya dari kueri yang sama kemudian mengembalikan hasil baru. Jika `enableResultCacheWithTTL` dilampirkan ke kueri berikutnya, hasil baru di-cache dengan TTL yang ditentukannya.

Petunjuk kueri Gremlin **`invalidateResultCacheKey`**

Petunjuk `invalidateResultCacheKey` kueri dapat mengambil `false` nilai `true` atau. `true`Nilai menyebabkan hasil cache untuk kueri yang `invalidateResultCacheKey` dilampirkan akan dihapus. Misalnya, contoh berikut menyebabkan hasil cache `g.V().has('genre','drama').in('likes')` untuk kunci kueri dihapus:

```
g.with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Contoh kueri di atas tidak menyebabkan hasil barunya di-cache. Anda dapat menyertakan `enableResultCache` (atau`enableResultCacheWithTTL`) dalam kueri yang sama jika Anda ingin menyimpan hasil baru setelah menghapus hasil cache yang ada:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Petunjuk kueri Gremlin **`invalidateResultCache`**

Petunjuk `invalidateResultCache` kueri dapat mengambil `false` nilai `true` atau. `true`Nilai menyebabkan semua hasil dalam cache hasil dihapus. Contoh:

```
g.with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Contoh query di atas tidak menyebabkan hasilnya di-cache. Anda dapat menyertakan `enableResultCache` (atau`enableResultCacheWithTTL`) dalam kueri yang sama jika Anda ingin menyimpan hasil baru setelah benar-benar menghapus cache yang ada:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Petunjuk kueri Gremlin **`numResultsCached`**

Petunjuk `numResultsCached` kueri hanya dapat digunakan dengan kueri yang berisi`iterate()`, dan menentukan jumlah maksimum hasil untuk cache untuk kueri yang dilampirkan. Perhatikan bahwa hasil yang di-cache saat `numResultsCached` ada tidak dikembalikan, hanya di-cache.

Misalnya, kueri berikut menetapkan bahwa hingga 100 hasilnya harus di-cache, tetapi tidak ada hasil cache yang dikembalikan:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').iterate()
```

Anda kemudian dapat menggunakan kueri seperti berikut ini untuk mengambil rentang hasil cache (di sini, sepuluh pertama):

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').range(0, 10)
```

Petunjuk kueri Gremlin **`noCacheExceptions`**

Petunjuk `noCacheExceptions` kueri dapat mengambil `false` nilai `true` atau. `true`Nilai menyebabkan pengecualian apa pun yang terkait dengan cache hasil ditekan. Contoh:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#noCacheExceptions', true)
 .V().has('genre','drama').in('likes')
```

Secara khusus, ini menekan`QueryLimitExceededException`, yang dinaikkan jika hasil kueri terlalu besar untuk dimasukkan ke dalam cache hasil.

API status kueri Gremlin

Untuk mendapatkan status kueri Gremlin, gunakan HTTP `GET` atau `POST` untuk membuat permintaan ke titik akhir `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin/status`.

Parameter permintaan status kueri Gremlin

- queryId (opsional) - ID dari kueri Gremlin yang berjalan. Hanya menampilkan status kueri yang ditentukan.
- includeWaiting (opsional) - Mengembalikan status semua kueri yang menunggu.

  Biasanya, hanya menjalankan kueri yang disertakan dalam respons, tetapi ketika parameter `includeWaiting` ditentukan, status semua kueri yang menunggu juga dikembalikan.

Sintaks respons status kueri Gremlin

```
{
  "acceptedQueryCount": integer,
  "runningQueryCount": integer,
  "queries": [
    {
      "queryId":"guid",
      "queryEvalStats":
        {
          "waited": integer,
          "elapsed": integer,
          "cancelled": boolean
        },
      "queryString": "string"
    }
  ]
}
```

Nilai respons status kueri Gremlin

- acceptedQueryCount— Jumlah kueri yang telah diterima tetapi belum selesai, termasuk kueri dalam antrian.
- runningQueryCount— Jumlah kueri Gremlin yang sedang berjalan.
- queries — Daftar kueri Gremlin saat ini.

- queryId — id GUID untuk kueri. Neptune secara otomatis memberikan nilai ID ini ke setiap kueri, atau Anda juga dapat menetapkan ID Anda sendiri (lihat Menyuntikkan ID Kustom Ke Dalam Gremlin Neptune atau Kueri SPARQL).
- queryEvalStats— Statistik untuk kueri ini.
- subqueries — Jumlah subqueries dalam kueri ini.
- elapsed — Jumlah milidetik kueri telah berjalan sejauh ini.
- cancelled — True menunjukkan bahwa kueri dibatalkan.
- queryString — Query yang dikirimkan. Ini dipotong menjadi 1024 karakter jika lebih panjang dari itu.
- menunggu - Menunjukkan berapa lama kueri menunggu, dalam milidetik.

Contoh status kueri Gremlin

Berikut ini adalah contoh perintah status menggunakan `curl` dan `GET` HTTP.

```
curl https://your-neptune-endpoint:port/gremlin/status
```

Output ini menunjukkan satu kueri yang berjalan.

```
{
  "acceptedQueryCount":9,
  "runningQueryCount":1,
  "queries": [
    {
      "queryId":"fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
      "queryEvalStats":
        {
          "waited": 0,
          "elapsed": 23,
          "cancelled": false
        },
      "queryString": "g.V().out().count()"
    }
  ]
}
```

Pembatalan kueri Gremlin

Untuk mendapatkan status kueri Gremlin, gunakan HTTP `GET` atau `POST` untuk membuat permintaan ke titik akhir `https://your-neptune-endpoint:port/gremlin/status`.

Parameter permintaan pembatalan kueri Gremlin

- cancelQuery — Diperlukan untuk pembatalan. Parameter ini tidak memiliki nilai yang sesuai.
- queryId — ID dari kueri Gremlin yang sedang berjalan untuk dibatalkan.

Contoh pembatalan kueri Gremlin

Berikut ini adalah contoh perintah `curl` untuk membatalkan kueri.

```
curl https://your-neptune-endpoint:port/gremlin/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=fb34cd3e-f37c-4d12-9cf2-03bb741bf54f"
```

Keberhasilan pembatalan kembali HTTP `200` OK.

Support untuk sesi berbasis skrip Gremlin

Anda dapat menggunakan sesi Gremlin dengan transaksi implisit di Amazon Neptune. Untuk informasi tentang sesi Gremlin, lihat Mempertimbangkan Sesi dalam dokumentasi TinkerPop Apache. Bagian di bawah ini menjelaskan cara menggunakan sesi Gremlin dengan Java.

> **Important**
>
> Saat ini, waktu terlama Neptunus dapat menjaga sesi berbasis skrip terbuka adalah 10 menit. Jika Anda tidak menutup sesi sebelum itu, waktu sesi habis dan semua yang ada di dalamnya di-rollback.

Topik

- Sesi Gremlin pada konsol Gremlin
- Sesi Gremlin dalam Varian Bahasa Gremlin

Sesi Gremlin pada konsol Gremlin

Jika Anda membuat koneksi jauh pada Konsol Gremlin tanpa parameter `session`, koneksi jauh dibuat di mode sessionless. Dalam mode ini, setiap permintaan yang diserahkan ke server diperlakukan sebagai transaksi lengkap dalam permintaan itu sendiri, dan tidak ada status yang disimpan antara permintaan. Jika permintaan gagal, hanya permintaan tersebut yang di-rollback.

Jika Anda membuat koneksi jarak jauh yang menggunakan `session` parameter, Anda membuat sesi berbasis skrip yang berlangsung hingga Anda menutup koneksi jarak jauh. Setiap sesi diidentifikasi oleh UUID unik yang dihasilkan konsol dan dikembalikan kepada Anda.

Berikut ini adalah contoh satu panggilan konsol yang menciptakan sesi. Setelah kueri dikirimkan, panggilan lain menutup sesi dan melakukan kueri.

> **Note**
>
> Klien Gremlin harus selalu ditutup untuk melepaskan sumber daya sisi server.

```
gremlin> :remote connect tinkerpop.server conf/neptune-remote.yaml session
  . . .
  . . .
gremlin> :remote close
```

Untuk informasi dan contoh selengkapnya, lihat Sesi dalam TinkerPop dokumentasi.

Semua pertanyaan yang Anda jalankan selama sesi membentuk satu transaksi yang tidak dilakukan sampai semua permintaan berhasil dan Anda menutup sambungan jarak jauh. Jika kueri gagal, atau jika Anda tidak menutup koneksi dalam masa sesi maksimum yang Neptune dukung, transaksi sesi tidak dilakukan, dan semua kueri di dalamnya di-rollback.

Sesi Gremlin dalam Varian Bahasa Gremlin

Dalam varian bahasa Gremlin (GLV), Anda perlu membuat objek `SessionedClient` untuk mengeluarkan beberapa kueri dalam satu transaksi, seperti dalam contoh berikut.

```
try {                              // line 1
  Cluster cluster = Cluster.open();                  // line 2
  Client client = cluster.connect("sessionName");    // line 3
```

```
    ...
    ...
} finally {
  // Always close. If there are no errors, the transaction is committed; otherwise,
 it's rolled back.
  client.close();
}
```

Baris 3 pada contoh sebelumnya membuat `SessionedClient` objek sesuai dengan opsi konfigurasi yang ditetapkan untuk cluster yang dimaksud. *`sessionName`*String yang Anda berikan ke metode connect menjadi nama unik sesi. Untuk menghindari tabrakan, gunakan UUID untuk nama tersebut.

Klien mulai transaksi sesi ketika diinisialisasi. Semua kueri yang Anda jalankan selama bentuk sesi dilakukan hanya ketika Anda memanggil `client.close( )`. Sekali lagi, jika kueri gagal, atau jika Anda tidak menutup koneksi dalam masa sesi maksimum yang Neptune dukung, transaksi sesi gagal, dan semua kueri di dalamnya di-rollback.

> **Note**
>
> Klien Gremlin harus selalu ditutup untuk melepaskan sumber daya sisi server.

```
GraphTraversalSource g = traversal().withRemote(conn);

Transaction tx = g.tx();

// Spawn a GraphTraversalSource from the Transaction.
// Traversals spawned from gtx are executed within a single transaction.
GraphTraversalSource gtx = tx.begin();
try {
  gtx.addV('person').iterate();
  gtx.addV('software').iterate();

  tx.commit();
} finally {
    if (tx.isOpen()) {
        tx.rollback();
    }
}
```

Transaksi Gremlin di Neptunus

Ada beberapa konteks di mana transaksi Gremlin dijalankan. Saat bekerja dengan Gremlin, penting untuk memahami konteks tempat Anda bekerja dan apa implikasinya:

- **`Script-based`**— Permintaan dibuat menggunakan string Gremlin berbasis teks, seperti ini:
  - Menggunakan driver Java dan `Client.submit(`*`string`*`)`
  - Menggunakan konsol Gremlin dan. `:remote connect`
  - Menggunakan HTTP API.
- **`Bytecode-based`**— Permintaan dibuat menggunakan bytecode Gremlin serial khas Gremlin Language Variants (GLV).

  Misalnya, menggunakan driver Java,`g = traversal().withRemote(...)`.

Untuk salah satu konteks di atas, ada konteks tambahan dari permintaan yang dikirim sebagai tanpa sesi atau terikat pada sesi.

> **Note**
>
> Transaksi Gremlin harus selalu dilakukan atau dibatalkan, sehingga sumber daya sisi server dapat dilepaskan. Jika terjadi kesalahan selama transaksi, penting untuk mencoba kembali seluruh transaksi dan bukan hanya permintaan tertentu yang gagal.

Permintaan tanpa sesi

Ketika sessionless, permintaan setara dengan satu transaksi.

Untuk skrip, implikasinya adalah bahwa satu atau lebih pernyataan Gremlin yang dikirim dalam satu permintaan akan melakukan atau mengembalikan sebagai satu transaksi. Contoh:

```
Cluster cluster = Cluster.open();
Client client = cluster.connect(); // sessionless
// 3 vertex additions in one request/transaction:
client.submit("g.addV();g.addV();g.addV()").all().get();
```

Untuk bytecode, permintaan tanpa sesi dibuat untuk setiap traversal yang muncul dan dieksekusi dari: g

```
GraphTraversalSource g = traversal().withRemote(...);

// 3 vertex additions in three individual requests/transactions:
g.addV().iterate();
g.addV().iterate();
g.addV().iterate();

// 3 vertex additions in one single request/transaction:
g.addV().addV().addV().iterate();
```

Permintaan terikat pada sesi

Ketika terikat ke sesi, beberapa permintaan dapat diterapkan dalam konteks satu transaksi.

Untuk skrip, implikasinya adalah bahwa tidak perlu menggabungkan semua operasi grafik menjadi satu nilai string yang disematkan:

```
Cluster cluster = Cluster.open();
Client client = cluster.connect(sessionName); // session
try {
    // 3 vertex additions in one request/transaction:
    client.submit("g.addV();g.addV();g.addV()").all().get();
} finally {
    client.close();
}

try {
    // 3 vertex additions in three requests, but one transaction:
    client.submit("g.addV()").all().get(); // starts a new transaction with the same
 sessionName
    client.submit("g.addV()").all().get();
    client.submit("g.addV()").all().get();
} finally {
    client.close();
}
```

Untuk bytecode, setelah TinkerPop `3.5.x`, transaksi dapat dikontrol secara eksplisit dan sesi dikelola secara transparan. Gremlin Language Variants (GLV) mendukung `tx()` sintaks Gremlin atau transaksi sebagai berikut: `commit()` `rollback()`

```
GraphTraversalSource g = traversal().withRemote(conn);
```

```
Transaction tx = g.tx();

// Spawn a GraphTraversalSource from the Transaction.
// Traversals spawned from gtx are executed within a single transaction.
GraphTraversalSource gtx = tx.begin();
try {
    gtx.addV('person').iterate();
    gtx.addV('software').iterate();

    tx.commit();
} finally {
    if (tx.isOpen()) {
        tx.rollback();
    }
}
```

Meskipun contoh di atas ditulis dalam Java, Anda juga dapat menggunakan `tx()` sintaks ini dalam Python, Javascript dan .NET.

> **Warning**
>
> Kueri hanya-baca tanpa sesi dijalankan di bawah isolasi SNAPSHOT, tetapi kueri hanya-baca yang dijalankan dalam transaksi eksplisit dijalankan di bawah isolasi SERIALIZABLE. Kueri hanya-baca yang dijalankan di bawah `SERIALIZABLE` isolasi menimbulkan overhead yang lebih tinggi dan dapat memblokir atau diblokir oleh penulisan bersamaan, tidak seperti yang dijalankan di bawah isolasi. `SNAPSHOT`

Menggunakan API Gremlin dengan Amazon Neptune

> **Note**
>
> Amazon Neptune tidak mendukung properti `bindings`.

Permintaan Gremlin HTTPS semua menggunakan titik akhir tunggal: `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin`. Semua koneksi Neptune harus menggunakan HTTPS.

Anda dapat menghubungkan Konsol Gremlin ke grafik Neptunus secara langsung. WebSockets

Untuk informasi selengkapnya tentang menghubungkan ke titik akhir Gremlin, lihat Mengakses grafik Neptune dengan Gremlin.

Implementasi Amazon Neptune untuk Gremlin memiliki detail dan perbedaan spesifik yang perlu Anda pertimbangkan. Untuk informasi selengkapnya, lihat Kepatuhan standar Gremlin di Amazon Neptune.

Untuk informasi tentang bahasa Gremlin dan traversal, lihat The Traversal dalam dokumentasi Apache. TinkerPop

Hasil kueri cache di Amazon Neptunus Gremlin

Amazon Neptunus mendukung cache hasil untuk kueri Gremlin.

Anda dapat mengaktifkan cache hasil kueri dan kemudian menggunakan petunjuk kueri untuk menyimpan hasil kueri hanya-baca Gremlin.

Setiap menjalankan ulang kueri kemudian mengambil hasil yang di-cache dengan latensi rendah dan tanpa I/O biaya, selama masih dalam cache. Ini berfungsi untuk kueri yang dikirimkan baik pada titik akhir HTTP dan menggunakan Websockets, baik sebagai kode byte atau dalam bentuk string.

> **Note**
>
> Kueri yang dikirim ke titik akhir profil tidak di-cache bahkan ketika cache kueri diaktifkan.

Anda dapat mengontrol bagaimana cache hasil kueri Neptunus berperilaku dalam beberapa cara. Contoh:

- Anda bisa mendapatkan hasil cache paginasi, dalam blok.
- Anda dapat menentukan time-to-live (TTL) untuk kueri tertentu.
- Anda dapat menghapus cache untuk kueri tertentu.
- Anda dapat menghapus seluruh cache.
- Anda dapat mengatur untuk diberi tahu jika hasil melebihi ukuran cache.

Cache dipertahankan menggunakan kebijakan least-recently-used (LRU), yang berarti bahwa setelah ruang yang dialokasikan ke cache penuh, least-recently-used hasilnya dihapus untuk memberi ruang ketika hasil baru di-cache.

> **Important**
>
> Cache hasil kueri tidak tersedia pada `t3.medium` atau jenis `t4.medium` instance.

Mengaktifkan cache hasil kueri di Neptunus

Cache hasil kueri dapat diaktifkan di semua instance dalam cluster atau per-instance. Untuk mengaktifkan cache hasil pada semua instance dalam sebuah cluster, atur `neptune_result_cache` parameter di cluster `cluster-parameter-group` ke`1`. Untuk mengaktifkan ini pada instance tertentu, atur `neptune_result_cache` parameter dalam instance `instance-parameter-group` ke`1`. Pengaturan grup parameter cluster akan mengganti nilai grup parameter instance.

Restart diperlukan pada setiap instance yang terpengaruh untuk pengaturan parameter cache hasil yang akan diterapkan. Meskipun Anda dapat mengaktifkan cache hasil di semua instance dalam cluster melalui`cluster-parameter-group`, setiap instance mempertahankan cache-nya sendiri. Fitur cache hasil kueri bukan cache di seluruh cluster.

Setelah cache hasil diaktifkan, Neptunus menyisihkan sebagian memori saat ini untuk hasil kueri caching. Semakin besar jenis instans yang Anda gunakan dan semakin banyak memori yang tersedia, semakin banyak memori yang disisihkan Neptunus untuk cache.

Jika memori cache hasil terisi, Neptunus secara otomatis least-recently-used menjatuhkan (LRU) hasil cache untuk memberi jalan bagi yang baru.

Anda dapat memeriksa status cache hasil saat ini menggunakan Status instans perintah.

Menggunakan petunjuk untuk menyimpan hasil kueri cache

Setelah cache hasil kueri diaktifkan, Anda menggunakan petunjuk kueri untuk mengontrol cache kueri. Semua contoh di bawah ini berlaku untuk traversal kueri yang sama, yaitu:

```
g.V().has('genre','drama').in('likes')
```

Menggunakan **`enableResultCache`**

Dengan cache hasil kueri diaktifkan, Anda dapat menyimpan hasil kueri Gremlin menggunakan petunjuk `enableResultCache` kueri, sebagai berikut:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Neptunus kemudian mengembalikan hasil kueri kepada Anda, dan juga menyimpannya dalam cache. Kemudian, Anda dapat mengakses hasil cache dengan mengeluarkan kueri yang persis sama lagi:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Kunci cache yang mengidentifikasi hasil cache adalah string kueri itu sendiri, yaitu:

```
g.V().has('genre','drama').in('likes')
```

Menggunakan **`enableResultCacheWithTTL`**

Anda dapat menentukan berapa lama hasil kueri harus di-cache dengan menggunakan petunjuk `enableResultCacheWithTTL` kueri. Misalnya, kueri berikut menentukan bahwa hasil kueri akan kedaluwarsa setelah 120 detik:

```
g.with('Neptune#enableResultCacheWithTTL', 120)
 .V().has('genre','drama').in('likes')
```

Sekali lagi, kunci cache yang mengidentifikasi hasil cache adalah string kueri dasar:

```
g.V().has('genre','drama').in('likes')
```

Dan lagi, Anda dapat mengakses hasil cache menggunakan string kueri itu dengan petunjuk `enableResultCache` kueri:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Jika 120 detik atau lebih telah berlalu sejak hasilnya di-cache, kueri itu akan mengembalikan hasil baru, dan menyimpannya dalam cache, tanpa ada time-to-live.

Anda juga dapat mengakses hasil cache dengan mengeluarkan kueri yang sama lagi dengan petunjuk `enableResultCacheWithTTL` kueri. Contoh:

```
g.with('Neptune#enableResultCacheWithTTL', 140)
```

```
  .V().has('genre','drama').in('likes')
```

Hingga 120 detik telah berlalu (yaitu, TTL saat ini berlaku), kueri baru ini menggunakan petunjuk `enableResultCacheWithTTL` kueri mengembalikan hasil yang di-cache. Setelah 120 detik, itu akan mengembalikan hasil baru dan menyimpannya dalam cache dengan time-to-live 140 detik.

> **Note**
>
> Jika hasil untuk kunci kueri sudah di-cache, maka kunci kueri yang sama dengan `enableResultCacheWithTTL` tidak menghasilkan hasil baru dan tidak berpengaruh pada hasil yang time-to-live saat ini di-cache.
>
> - Jika hasil sebelumnya di-cache menggunakan`enableResultCache`, cache harus dibersihkan terlebih dahulu sebelum `enableResultCacheWithTTL` menghasilkan hasil baru dan menyimpannya untuk TTL yang ditentukannya.
> - Jika hasil sebelumnya di-cache menggunakan`enableResultCachewithTTL`, TTL sebelumnya harus kedaluwarsa terlebih dahulu sebelum `enableResultCacheWithTTL` menghasilkan hasil baru dan menyimpannya untuk TTL yang ditentukannya.

Menggunakan **`invalidateResultCacheKey`**

Anda dapat menggunakan petunjuk `invalidateResultCacheKey` kueri untuk menghapus hasil cache untuk satu kueri tertentu. Contoh:

```
g.with('Neptune#invalidateResultCacheKey', true)
  .V().has('genre','drama').in('likes')
```

Kueri itu menghapus cache untuk kunci kuerig`.V().has('genre','drama').in('likes')`, dan mengembalikan hasil baru untuk kueri itu.

Anda juga dapat menggabungkan `invalidateResultCacheKey` dengan `enableResultCache` atau`enableResultCacheWithTTL`. Misalnya, kueri berikut menghapus hasil cache saat ini, menyimpan hasil baru, dan mengembalikannya:

```
g.with('Neptune#enableResultCache', true)
  .with('Neptune#invalidateResultCacheKey', true)
  .V().has('genre','drama').in('likes')
```

Menggunakan **`invalidateResultCache`**

Anda dapat menggunakan petunjuk `invalidateResultCache` kueri untuk menghapus semua hasil cache di cache hasil kueri. Contoh:

```
g.with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Kueri itu menghapus seluruh cache hasil dan mengembalikan hasil baru untuk kueri.

Anda juga dapat menggabungkan `invalidateResultCache` dengan `enableResultCache` atau`enableResultCacheWithTTL`. Misalnya, kueri berikut menghapus seluruh cache hasil, menyimpan hasil baru untuk kueri ini, dan mengembalikannya:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Paginasi hasil kueri yang di-cache

Misalkan Anda telah melakukan cache sejumlah besar hasil seperti ini:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Sekarang misalkan Anda mengeluarkan kueri rentang berikut:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes').range(0,10)
```

Neptunus pertama kali mencari kunci cache penuh, yaitu. `g.V().has('genre','drama').in('likes').range(0,10)` Jika kunci itu tidak ada, Neptunus selanjutnya melihat apakah ada kunci untuk string kueri itu tanpa rentang (yaitu). `g.V().has('genre','drama').in('likes')` Ketika menemukan kunci itu, Neptunus kemudian mengambil sepuluh hasil pertama dari cache-nya, seperti yang ditentukan oleh rentang.

> **Note**
>
> Jika Anda menggunakan petunjuk `invalidateResultCacheKey` kueri dengan kueri yang memiliki rentang di akhir, Neptunus menghapus cache untuk kueri tanpa rentang jika tidak menemukan kecocokan yang tepat untuk kueri dengan rentang tersebut.

Menggunakan **`numResultsCached`** dengan **`.iterate()`**

Dengan menggunakan petunjuk `numResultsCached` kueri, Anda dapat mengisi cache hasil tanpa mengembalikan semua hasil yang di-cache, yang dapat berguna saat Anda memilih untuk membuat halaman sejumlah besar hasil.

Petunjuk `numResultsCached` kueri hanya berfungsi dengan kueri yang diakhiri dengan. `iterate()`

Misalnya, jika Anda ingin menyimpan 50 hasil pertama dari kueri sampel:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').iterate()
```

Dalam hal ini kunci kueri dalam cache adalah:`g.with("Neptune#numResultsCached", 50).V().has('genre','drama').in('likes')`. Anda sekarang dapat mengambil sepuluh pertama dari hasil cache dengan kueri ini:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').range(0, 10)
```

Dan, Anda dapat mengambil sepuluh hasil berikutnya dari kueri sebagai berikut:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').range(10, 20)
```

Jangan lupa sertakan `numResultsCached` petunjuknya! Ini adalah bagian penting dari kunci kueri dan karena itu harus ada untuk mengakses hasil cache.

Beberapa hal yang perlu diingat saat menggunakan **`numResultsCached`**

- Nomor yang Anda berikan **`numResultsCached`** diterapkan di akhir kueri. Ini berarti, misalnya, bahwa kueri berikut sebenarnya cache menghasilkan rentang`(1000, 1500)`:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).iterate()
```

- Nomor yang Anda berikan **`numResultsCached`** menentukan jumlah maksimum hasil untuk cache. Ini berarti, misalnya, bahwa kueri berikut sebenarnya cache menghasilkan rentang`(1000, 2000)`:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 100000)
 .V().range(1000, 2000).iterate()
```

- Hasil yang di-cache oleh kueri yang diakhiri dengan **`.range().iterate()`** memiliki jangkauannya sendiri. Misalnya, Anda menyimpan hasil cache menggunakan kueri seperti ini:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).iterate()
```

  Untuk mengambil 100 hasil pertama dari cache, Anda akan menulis kueri seperti ini:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).range(0, 100)
```

  Seratus hasil itu akan setara dengan hasil dari kueri dasar dalam kisaran tersebut`(1000, 1100)`.

Kunci cache kueri yang digunakan untuk menemukan hasil cache

Setelah hasil kueri di-cache, kueri berikutnya dengan kunci cache kueri yang sama mengambil hasil dari cache daripada menghasilkan yang baru. Kunci cache kueri dari kueri dievaluasi sebagai berikut:

1. Semua petunjuk kueri terkait cache diabaikan, kecuali untuk. `numResultsCached`
2. `iterate()`Langkah terakhir diabaikan.

3. Sisa kueri diurutkan sesuai dengan representasi kode byte.

String yang dihasilkan dicocokkan dengan indeks hasil kueri yang sudah ada di cache untuk menentukan apakah ada cache hit untuk kueri.

Misalnya, ambil kueri ini:

```
g.withSideEffect('Neptune#typePromotion', false).with("Neptune#enableResultCache",
 true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').iterate()
```

Ini akan disimpan sebagai versi byte-code ini:

```
g.withSideEffect('Neptune#typePromotion', false)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes')
```

Pengecualian terkait dengan cache hasil

Jika hasil kueri yang Anda coba cache terlalu besar untuk dimasukkan ke dalam memori cache bahkan setelah menghapus semua yang sebelumnya di-cache, Neptunus menimbulkan kesalahan. `QueryLimitExceededException` Tidak ada hasil yang dikembalikan, dan pengecualian menghasilkan pesan galat berikut:

```
The result size is larger than the allocated cache,
      please refer to results cache best practices for options to rerun the query.
```

Anda dapat menekan pesan ini menggunakan petunjuk `noCacheExceptions` kueri, sebagai berikut:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#noCacheExceptions', true)
 .V().has('genre','drama').in('likes')
```

Membuat peningkatan yang efisien dengan **`mergeV()`** Gremlin dan langkah-langkah **`mergeE()`**

Upsert (atau sisipan bersyarat) menggunakan kembali simpul atau tepi jika sudah ada, atau membuatnya jika tidak. Upserts yang efisien dapat membuat perbedaan yang signifikan dalam kinerja kueri Gremlin.

Upserts memungkinkan Anda untuk menulis operasi penyisipan idempoten: tidak peduli berapa kali Anda menjalankan operasi seperti itu, hasil keseluruhannya sama. Ini berguna dalam skenario penulisan yang sangat bersamaan di mana modifikasi bersamaan pada bagian grafik yang sama dapat memaksa satu atau lebih transaksi untuk memutar kembali dengan a`ConcurrentModificationException`, sehingga memerlukan percobaan ulang.

Misalnya, kueri berikut meningkatkan simpul dengan menggunakan yang disediakan `Map` untuk pertama kali mencoba menemukan simpul dengan a of. `T.id "v-1"` Jika simpul itu ditemukan maka dikembalikan. Jika tidak ditemukan maka simpul dengan itu `id` dan properti dibuat melalui `onCreate` klausa.

```
g.mergeV([(id):'v-1']).
  option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org'])
```

Batching upserts untuk meningkatkan throughput

Untuk skenario penulisan throughput tinggi, Anda dapat berantai `mergeV()` dan `mergeE()` melangkah bersama untuk meningkatkan simpul dan tepi dalam batch. Batching mengurangi overhead transaksional untuk menaikkan sejumlah besar simpul dan tepi. Anda kemudian dapat lebih meningkatkan throughput dengan meningkatkan permintaan batch secara paralel menggunakan beberapa klien.

Sebagai aturan praktis, kami merekomendasikan untuk meningkatkan sekitar 200 catatan per permintaan batch. Rekaman adalah satu titik atau label tepi atau properti. Sebuah simpul dengan label tunggal dan 4 properti, misalnya, membuat 5 catatan. Tepi dengan label dan properti tunggal menciptakan 2 catatan. Jika Anda ingin meningkatkan kumpulan simpul, masing-masing dengan label tunggal dan 4 properti, Anda harus mulai dengan ukuran batch 40, karena. `200 / (1 + 4) = 40`

Anda dapat bereksperimen dengan ukuran batch. 200 catatan per batch adalah titik awal yang baik, tetapi ukuran batch yang ideal mungkin lebih tinggi atau lebih rendah tergantung pada beban kerja

Anda. Perhatikan, bagaimanapun, bahwa Neptunus dapat membatasi jumlah keseluruhan langkah Gremlin per permintaan. Batas ini tidak didokumentasikan, tetapi untuk berada di sisi yang aman, cobalah untuk memastikan bahwa permintaan Anda mengandung tidak lebih dari 1.500 langkah Gremlin. Neptunus dapat menolak permintaan batch besar dengan lebih dari 1.500 langkah.

Untuk meningkatkan throughput, Anda dapat meningkatkan batch secara paralel menggunakan beberapa klien (lihat). Membuat Penulisan Gremlin Multithreaded yang Efisien Jumlah klien harus sama dengan jumlah thread pekerja pada instance penulis Neptunus Anda, yang biasanya 2 x jumlah CPUs v di server. Misalnya, sebuah `r5.8xlarge` instance memiliki 32 v CPUs dan 64 thread pekerja. Untuk skenario penulisan throughput tinggi menggunakan`r5.8xlarge`, Anda akan menggunakan 64 klien yang menulis batch upserts ke Neptunus secara paralel.

Setiap klien harus mengirimkan permintaan batch dan menunggu permintaan selesai sebelum mengirimkan permintaan lain. Meskipun beberapa klien berjalan secara paralel, setiap klien individu mengirimkan permintaan secara serial. Ini memastikan bahwa server disuplai dengan aliran permintaan yang stabil yang menempati semua thread pekerja tanpa membanjiri antrian permintaan sisi server (lihat). Mengukur instans DB dalam sebuah klaster Neptune DB

Cobalah untuk menghindari langkah-langkah yang menghasilkan banyak pelintas

Ketika langkah Gremlin dijalankan, dibutuhkan traverser masuk, dan memancarkan satu atau lebih traverser keluaran. Jumlah pelintas yang dipancarkan oleh satu langkah menentukan berapa kali langkah berikutnya dijalankan.

Biasanya, ketika melakukan operasi batch Anda ingin setiap operasi, seperti upsert vertex A, untuk mengeksekusi sekali, sehingga urutan operasi terlihat seperti ini: upsert vertex A, kemudian upsert vertex B, kemudian upsert vertex C, dan seterusnya. Selama sebuah langkah membuat atau memodifikasi hanya satu elemen, ia hanya memancarkan satu traverser, dan langkah-langkah yang mewakili operasi berikutnya dijalankan hanya sekali. Jika, di sisi lain, sebuah operasi membuat atau memodifikasi lebih dari satu elemen, ia memancarkan beberapa pelintas, yang pada gilirannya menyebabkan langkah-langkah selanjutnya dieksekusi beberapa kali, sekali per traverser yang dipancarkan. Hal ini dapat mengakibatkan database melakukan pekerjaan tambahan yang tidak perlu, dan dalam beberapa kasus dapat mengakibatkan penciptaan simpul tambahan yang tidak diinginkan, tepi atau nilai properti.

Contoh bagaimana hal-hal bisa salah adalah dengan kueri sepertig`.V().addV()`. Kueri sederhana ini menambahkan simpul untuk setiap simpul yang ditemukan dalam grafik, karena `V()` memancarkan traverser untuk setiap simpul dalam grafik dan masing-masing pelintas tersebut memicu panggilan ke. `addV()`

Lihat Mencampur upserts dan sisipan cara menangani operasi yang dapat memancarkan banyak pelintas.

Menaikkan simpul

`mergeV()`Langkah ini dirancang khusus untuk menaikkan simpul. Dibutuhkan sebagai argumen a `Map` yang mewakili elemen untuk mencocokkan simpul yang ada dalam grafik, dan jika elemen tidak ditemukan, menggunakannya `Map` untuk membuat simpul baru. Langkah ini juga memungkinkan Anda untuk mengubah perilaku jika terjadi penciptaan atau kecocokan, di mana `option()` modulator dapat diterapkan dengan `Merge.onCreate` dan `Merge.onMatch` token untuk mengontrol perilaku masing-masing. Lihat Dokumentasi TinkerPop Referensi untuk informasi lebih lanjut tentang cara menggunakan langkah ini.

Anda dapat menggunakan ID simpul untuk menentukan apakah ada simpul tertentu. Ini adalah pendekatan yang lebih disukai, karena Neptunus mengoptimalkan upserts untuk kasus penggunaan yang sangat bersamaan. IDs Sebagai contoh, kueri berikut membuat simpul dengan ID simpul yang diberikan jika belum ada, atau menggunakannya kembali jika memang demikian:

```
g.mergeV([(T.id): 'v-1']).
    option(onCreate, [(T.label): 'PERSON', email: 'person-1@example.org', age: 21]).
    option(onMatch, [age: 22]).
  id()
```

Perhatikan bahwa kueri ini diakhiri dengan `id()` langkah. Meskipun tidak sepenuhnya diperlukan untuk tujuan meningkatkan simpul, `id()` langkah ke akhir kueri upsert memastikan bahwa server tidak membuat serial semua properti simpul kembali ke klien, yang membantu mengurangi biaya penguncian kueri.

Atau, Anda dapat menggunakan properti simpul untuk mengidentifikasi simpul:

```
g.mergeV([email: 'person-1@example.org']).
    option(onCreate, [(T.label): 'PERSON', age: 21]).
    option(onMatch, [age: 22]).
  id()
```

Jika memungkinkan, gunakan sendiri yang disediakan pengguna IDs untuk membuat simpul, dan gunakan ini IDs untuk menentukan apakah simpul ada selama operasi upsert. Ini memungkinkan Neptunus mengoptimalkan upserts. Upsert berbasis ID dapat secara signifikan lebih efisien daripada upsert berbasis properti ketika modifikasi bersamaan adalah umum.

Bagian atas simpul berantai

Anda dapat menghubungkan bagian atas simpul bersama-sama untuk menyisipkannya dalam batch:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org'))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org'))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-3')
                         .property('email', 'person-3@example.org'))
 .id()
```

Atau, Anda juga dapat menggunakan `mergeV()` sintaks ini:

```
g.mergeV([(T.id): 'v-1', (T.label): 'PERSON', email: 'person-1@example.org']).
  mergeV([(T.id): 'v-2', (T.label): 'PERSON', email: 'person-2@example.org']).
  mergeV([(T.id): 'v-3', (T.label): 'PERSON', email: 'person-3@example.org'])
```

Namun, karena bentuk kueri ini menyertakan elemen dalam kriteria penelusuran yang berlebihan untuk pencarian dasar`id`, itu tidak seefisien kueri sebelumnya.

Tepi yang menjulang

`mergeE()`Langkah ini dirancang khusus untuk menaikkan tepi. Dibutuhkan `Map` sebagai argumen yang mewakili elemen untuk mencocokkan tepi yang ada dalam grafik dan jika elemen tidak ditemukan, menggunakannya `Map` untuk membuat tepi baru. Langkah ini juga memungkinkan Anda untuk mengubah perilaku jika terjadi penciptaan atau kecocokan, di mana `option()` modulator dapat diterapkan dengan `Merge.onCreate` dan `Merge.onMatch` token untuk mengontrol perilaku masing-masing. Lihat Dokumentasi TinkerPop Referensi untuk informasi lebih lanjut tentang cara menggunakan langkah ini.

Anda dapat menggunakan tepi IDs untuk menaikkan tepi dengan cara yang sama Anda menaikkan simpul menggunakan simpul kustom. IDs Sekali lagi, ini adalah pendekatan yang disukai karena memungkinkan Neptunus untuk mengoptimalkan kueri. Misalnya, kueri berikut membuat tepi berdasarkan ID tepinya jika belum ada, atau menggunakannya kembali jika ada. Kueri juga menggunakan simpul `Direction.from` dan `Direction.to` simpul jika perlu membuat tepi baru: IDs

```
g.mergeE([(T.id): 'e-1']).
    option(onCreate, [(from): 'v-1', (to): 'v-2', weight: 1.0]).
    option(onMatch, [weight: 0.5]).
  id()
```

Perhatikan bahwa kueri ini diakhiri dengan `id()` langkah. Meskipun tidak sepenuhnya diperlukan untuk tujuan meningkatkan tepi, menambahkan `id()` langkah ke akhir kueri upsert memastikan bahwa server tidak membuat serial semua properti edge kembali ke klien, yang membantu mengurangi biaya penguncian kueri.

Banyak aplikasi menggunakan simpul khusus IDs, tetapi meninggalkan Neptunus untuk menghasilkan tepi. IDs Jika Anda tidak tahu ID tepi, tetapi Anda tahu `from` dan `to` simpulnya IDs, Anda dapat menggunakan kueri semacam ini untuk meningkatkan tepi:

```
g.mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  id()
```

Semua simpul yang direferensikan oleh `mergeE()` harus ada untuk langkah untuk membuat tepi.

Bagian atas tepi rantai

Seperti halnya vertex upserts, sangat mudah untuk menggabungkan `mergeE()` langkah-langkah bersama untuk permintaan batch:

```
g.mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  mergeE([(from): 'v-2', (to): 'v-3', (T.label): 'KNOWS']).
  mergeE([(from): 'v-3', (to): 'v-4', (T.label): 'KNOWS']).
  id()
```

Menggabungkan vertex dan edge upserts

Terkadang Anda mungkin ingin meningkatkan kedua simpul dan tepi yang menghubungkannya. Anda dapat mencampur contoh batch yang disajikan di sini. Contoh berikut meningkatkan 3 simpul dan 2 tepi:

```
g.mergeV([(id):'v-1']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org']).
  mergeV([(id):'v-2']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-2@example.org']).
  mergeV([(id):'v-3']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-3@example.org']).
  mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  mergeE([(from): 'v-2', (to): 'v-3', (T.label): 'KNOWS']).
 id()
```

Mencampur upserts dan sisipan

Terkadang Anda mungkin ingin meningkatkan kedua simpul dan tepi yang menghubungkannya. Anda dapat mencampur contoh batch yang disajikan di sini. Contoh berikut meningkatkan 3 simpul dan 2 tepi:

Upserts biasanya melanjutkan satu elemen pada satu waktu. Jika Anda tetap berpegang pada pola upsert yang disajikan di sini, setiap operasi upsert memancarkan satu traverser, yang menyebabkan operasi berikutnya dijalankan hanya sekali.

Namun, terkadang Anda mungkin ingin mencampur upserts dengan sisipan. Ini bisa terjadi, misalnya, jika Anda menggunakan tepi untuk mewakili contoh tindakan atau peristiwa. Permintaan mungkin menggunakan upserts untuk memastikan bahwa semua simpul yang diperlukan ada, dan kemudian menggunakan sisipan untuk menambahkan tepi. Dengan permintaan semacam ini, perhatikan potensi jumlah pelintas yang dipancarkan dari setiap operasi.

Perhatikan contoh berikut, yang mencampur upsert dan sisipan untuk menambahkan tepi yang mewakili peristiwa ke dalam grafik:

```
// Fully optimized, but inserts too many edges
g.mergeV([(id):'v-1']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org']).
  mergeV([(id):'v-2']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-2@example.org']).
```

```
  mergeV([(id):'v-3']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-3@example.org']).
  mergeV([(T.id): 'c-1', (T.label): 'CITY', name: 'city-1']).
  V('p-1', 'p-2').
  addE('FOLLOWED').to(V('p-1')).
  V('p-1', 'p-2', 'p-3').
  addE('VISITED').to(V('c-1')).
  id()
```

Kueri harus menyisipkan 5 tepi: 2 tepi DIIKUTI dan 3 tepi VISITED. Namun, kueri sebagai tertulis menyisipkan 8 tepi: 2 DIIKUTI dan 6 DIKUNJUNGI. Alasan untuk ini adalah bahwa operasi yang menyisipkan 2 tepi FOLLOW memancarkan 2 traversers, menyebabkan operasi penyisipan berikutnya, yang menyisipkan 3 tepi, dieksekusi dua kali.

Perbaikannya adalah menambahkan `fold()` langkah setelah setiap operasi yang berpotensi memancarkan lebih dari satu traverser:

```
g.mergeV([(T.id): 'v-1', (T.label): 'PERSON', email: 'person-1@example.org']).
  mergeV([(T.id): 'v-2', (T.label): 'PERSON', email: 'person-2@example.org']).
  mergeV([(T.id): 'v-3', (T.label): 'PERSON', email: 'person-3@example.org']).
  mergeV([(T.id): 'c-1', (T.label): 'CITY', name: 'city-1']).
  V('p-1', 'p-2').
  addE('FOLLOWED').
    to(V('p-1')).
  fold().
  V('p-1', 'p-2', 'p-3').
  addE('VISITED').
    to(V('c-1')).
  id()
```

Di sini kita telah memasukkan `fold()` langkah setelah operasi yang menyisipkan tepi DIIKUTI. Ini menghasilkan traverser tunggal, yang kemudian menyebabkan operasi berikutnya dieksekusi hanya sekali.

Kelemahan dari pendekatan ini adalah bahwa query sekarang tidak sepenuhnya dioptimalkan, karena tidak `fold()` dioptimalkan. Operasi insert yang mengikuti sekarang juga tidak `fold()` akan dioptimalkan.

Jika Anda perlu menggunakan `fold()` untuk mengurangi jumlah pelintas atas nama langkah-langkah selanjutnya, cobalah untuk memesan operasi Anda sehingga yang paling murah menempati bagian kueri yang tidak dioptimalkan.

Pengaturan Kardinalitas

Kardinalitas default untuk properti simpul di Neptunus diatur, yang berarti bahwa ketika menggunakan mergeV () nilai yang diberikan dalam peta semuanya akan diberikan kardinalitas itu. Untuk menggunakan kardinalitas tunggal, Anda harus eksplisit dalam penggunaannya. Mulai dari TinkerPop 3.7.0, ada sintaks baru yang memungkinkan kardinalitas diberikan sebagai bagian dari peta seperti yang ditunjukkan pada contoh berikut:

```
g.mergeV([(T.id): '1234']).
  option(onMatch, ['age': single(20), 'name': single('alice'), 'city': set('miami')])
```

Atau, Anda dapat menetapkan kardinalitas sebagai default untuk itu `option` sebagai berikut:

```
// age and name are set to single cardinality by default
g.mergeV([(T.id): '1234']).
  option(onMatch, ['age': 22, 'name': 'alice', 'city': set('boston')], single)
```

Ada lebih sedikit opsi untuk mengatur kardinalitas `mergeV()` sebelum versi 3.7.0. Pendekatan umum adalah kembali ke `property()` langkah sebagai berikut:

```
g.mergeV([(T.id): '1234']).
  option(onMatch, sideEffect(property(single,'age', 20).
  property(set,'city','miami')).constant([:]))
```

> **Note**
>
> Pendekatan ini hanya akan bekerja dengan `mergeV()` ketika digunakan dengan langkah awal. Oleh karena itu Anda tidak akan dapat berantai `mergeV()` dalam satu traversal karena yang pertama `mergeV()` setelah langkah awal yang menggunakan sintaks ini akan menghasilkan kesalahan jika traverser yang masuk menjadi elemen grafik. Dalam hal ini, Anda ingin memecah `mergeV()` panggilan Anda menjadi beberapa permintaan di mana masing-masing dapat menjadi langkah awal.

Membuat peningkatan Gremlin yang efisien dengan `fold()/coalesce()/unfold()`

Upsert (atau sisipan bersyarat) menggunakan kembali simpul atau tepi jika sudah ada, atau membuatnya jika tidak. Upserts yang efisien dapat membuat perbedaan yang signifikan dalam kinerja kueri Gremlin.

Halaman ini menunjukkan bagaimana menggunakan pola `fold()/coalesce()/unfold()` Gremlin untuk membuat upserts yang efisien. Namun, dengan rilis TinkerPop versi 3.6.x yang diperkenalkan di Neptunus dalam versi mesin 1.2.1.0, langkah baru `mergeV()` dan lebih disukai dalam banyak kasus. `mergeE()` `fold()/coalesce()/unfold()`Pola yang dijelaskan di sini mungkin masih berguna dalam beberapa situasi yang kompleks, tetapi dalam penggunaan umum `mergeV()` dan `mergeE()` jika Anda bisa, seperti yang dijelaskan dalamMembuat peningkatan yang efisien dengan `mergeV()` Gremlin dan langkah-langkah `mergeE()`.

Upserts memungkinkan Anda untuk menulis operasi penyisipan idempoten: tidak peduli berapa kali Anda menjalankan operasi seperti itu, hasil keseluruhannya sama. Ini berguna dalam skenario penulisan yang sangat bersamaan di mana modifikasi bersamaan pada bagian grafik yang sama dapat memaksa satu atau lebih transaksi untuk memutar kembali dengan a`ConcurrentModificationException`, sehingga memerlukan percobaan ulang.

Misalnya, kueri berikut meningkatkan simpul dengan terlebih dahulu mencari simpul yang ditentukan dalam kumpulan data, dan kemudian melipat hasilnya ke dalam daftar. Dalam traversal pertama yang diberikan ke `coalesce()` langkah, kueri kemudian membuka daftar ini. Jika daftar yang dibuka tidak kosong, hasilnya dipancarkan dari. `coalesce()` Namun, jika `unfold()` mengembalikan koleksi kosong karena simpul saat ini tidak ada, lanjutkan `coalesce()` untuk mengevaluasi traversal kedua yang telah disediakan, dan dalam traversal kedua ini kueri membuat simpul yang hilang.

```
g.V('v-1').fold()
          .coalesce(
             unfold(),
             addV('Person').property(id, 'v-1')
                           .property('email', 'person-1@example.org')
           )
```

Gunakan formulir yang dioptimalkan `coalesce()` untuk upserts

Neptunus dapat mengoptimalkan `fold().coalesce(unfold(), ...)` idiom untuk membuat pembaruan throughput tinggi, tetapi pengoptimalan ini hanya berfungsi jika kedua bagian

pengembalian baik simpul atau tepi `coalesce()` tetapi tidak ada yang lain. Jika Anda mencoba mengembalikan sesuatu yang berbeda, seperti properti, dari bagian mana pun`coalesce()`, pengoptimalan Neptunus tidak terjadi. Kueri mungkin berhasil, tetapi tidak akan berkinerja sebaik versi yang dioptimalkan, terutama terhadap kumpulan data besar.

Karena kueri upsert yang tidak dioptimalkan meningkatkan waktu eksekusi dan mengurangi throughput, ada baiknya menggunakan `explain` titik akhir Gremlin untuk menentukan apakah kueri upsert sepenuhnya dioptimalkan. Saat meninjau `explain` rencana, cari garis yang dimulai dengan `+ not converted into Neptune steps` dan`WARNING: >>`. Contoh:

```
+ not converted into Neptune steps: [FoldStep, CoalesceStep([[UnfoldStep],
 [AddEdgeSte...
WARNING: >> FoldStep << is not supported natively yet
```

Peringatan ini dapat membantu Anda mengidentifikasi bagian-bagian kueri yang mencegahnya dioptimalkan sepenuhnya.

Terkadang tidak mungkin untuk mengoptimalkan kueri sepenuhnya. Dalam situasi ini Anda harus mencoba meletakkan langkah-langkah yang tidak dapat dioptimalkan di akhir kueri, sehingga memungkinkan mesin untuk mengoptimalkan sebanyak mungkin langkah. Teknik ini digunakan dalam beberapa contoh batch upsert, di mana semua upsert yang dioptimalkan untuk satu set simpul atau tepi dilakukan sebelum modifikasi tambahan yang berpotensi tidak dioptimalkan diterapkan pada simpul atau tepi yang sama.

Batching upserts untuk meningkatkan throughput

Untuk skenario penulisan throughput tinggi, Anda dapat menggabungkan langkah-langkah upsert bersama-sama untuk meningkatkan simpul dan tepi dalam batch. Batching mengurangi overhead transaksional untuk menaikkan sejumlah besar simpul dan tepi. Anda kemudian dapat lebih meningkatkan throughput dengan meningkatkan permintaan batch secara paralel menggunakan beberapa klien.

Sebagai aturan praktis, kami merekomendasikan untuk meningkatkan sekitar 200 catatan per permintaan batch. Rekaman adalah satu titik atau label tepi atau properti. Sebuah simpul dengan label tunggal dan 4 properti, misalnya, membuat 5 catatan. Tepi dengan label dan properti tunggal menciptakan 2 catatan. Jika Anda ingin meningkatkan kumpulan simpul, masing-masing dengan label tunggal dan 4 properti, Anda harus mulai dengan ukuran batch 40, karena. `200 / (1 + 4) = 40`

Anda dapat bereksperimen dengan ukuran batch. 200 catatan per batch adalah titik awal yang baik, tetapi ukuran batch yang ideal mungkin lebih tinggi atau lebih rendah tergantung pada beban kerja Anda. Perhatikan, bagaimanapun, bahwa Neptunus dapat membatasi jumlah keseluruhan langkah Gremlin per permintaan. Batas ini tidak didokumentasikan, tetapi untuk berada di sisi yang aman cobalah untuk memastikan bahwa permintaan Anda mengandung tidak lebih dari 1500 langkah Gremlin. Neptunus dapat menolak permintaan batch besar dengan lebih dari 1500 langkah.

Untuk meningkatkan throughput, Anda dapat meningkatkan batch secara paralel menggunakan beberapa klien (lihat). Membuat Penulisan Gremlin Multithreaded yang Efisien Jumlah klien harus sama dengan jumlah thread pekerja pada instance penulis Neptunus Anda, yang biasanya 2 x jumlah CPUs v di server. Misalnya, sebuah `r5.8xlarge` instance memiliki 32 v CPUs dan 64 thread pekerja. Untuk skenario penulisan throughput tinggi menggunakan`r5.8xlarge`, Anda akan menggunakan 64 klien yang menulis batch upserts ke Neptunus secara paralel.

Setiap klien harus mengirimkan permintaan batch dan menunggu permintaan selesai sebelum mengirimkan permintaan lain. Meskipun beberapa klien berjalan secara paralel, setiap klien individu mengirimkan permintaan secara serial. Ini memastikan bahwa server disuplai dengan aliran permintaan yang stabil yang menempati semua thread pekerja tanpa membanjiri antrian permintaan sisi server (lihat). Mengukur instans DB dalam sebuah klaster Neptune DB

Cobalah untuk menghindari langkah-langkah yang menghasilkan banyak pelintas

Ketika langkah Gremlin dijalankan, dibutuhkan traverser masuk, dan memancarkan satu atau lebih traverser keluaran. Jumlah pelintas yang dipancarkan oleh satu langkah menentukan berapa kali langkah berikutnya dijalankan.

Biasanya, ketika melakukan operasi batch Anda ingin setiap operasi, seperti upsert vertex A, untuk mengeksekusi sekali, sehingga urutan operasi terlihat seperti ini: upsert vertex A, kemudian upsert vertex B, kemudian upsert vertex C, dan seterusnya. Selama sebuah langkah membuat atau memodifikasi hanya satu elemen, ia hanya memancarkan satu traverser, dan langkah-langkah yang mewakili operasi berikutnya dijalankan hanya sekali. Jika, di sisi lain, sebuah operasi membuat atau memodifikasi lebih dari satu elemen, ia memancarkan beberapa pelintas, yang pada gilirannya menyebabkan langkah-langkah selanjutnya dieksekusi beberapa kali, sekali per traverser yang dipancarkan. Hal ini dapat mengakibatkan database melakukan pekerjaan tambahan yang tidak perlu, dan dalam beberapa kasus dapat mengakibatkan penciptaan simpul tambahan yang tidak diinginkan, tepi atau nilai properti.

Contoh bagaimana hal-hal bisa salah adalah dengan kueri sepertig`.V().addV()`. Kueri sederhana ini menambahkan simpul untuk setiap simpul yang ditemukan dalam grafik, karena `V()`

memancarkan traverser untuk setiap simpul dalam grafik dan masing-masing pelintas tersebut memicu panggilan ke. `addV()`

Lihat Mencampur upserts dan sisipan cara menangani operasi yang dapat memancarkan banyak pelintas.

Menaikkan simpul

Anda dapat menggunakan ID simpul untuk menentukan apakah ada simpul yang sesuai. Ini adalah pendekatan yang lebih disukai, karena Neptunus mengoptimalkan upserts untuk kasus penggunaan yang sangat bersamaan. IDs Sebagai contoh, kueri berikut membuat simpul dengan ID simpul yang diberikan jika belum ada, atau menggunakannya kembali jika memang demikian:

```
g.V('v-1')
 .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-1')
                          .property('email', 'person-1@example.org'))
  .id()
```

Perhatikan bahwa kueri ini diakhiri dengan `id()` langkah. Meskipun tidak sepenuhnya diperlukan untuk tujuan meningkatkan simpul, menambahkan `id()` langkah ke akhir kueri upsert memastikan bahwa server tidak membuat serial semua properti simpul kembali ke klien, yang membantu mengurangi biaya penguncian kueri.

Atau, Anda dapat menggunakan properti simpul untuk menentukan apakah simpul itu ada:

```
g.V()
 .hasLabel('Person')
 .has('email', 'person-1@example.org')
 .fold()
 .coalesce(unfold(),
           addV('Person').property('email', 'person-1@example.org'))
 .id()
```

Jika memungkinkan, gunakan sendiri yang disediakan pengguna IDs untuk membuat simpul, dan gunakan ini IDs untuk menentukan apakah simpul ada selama operasi upsert. Ini memungkinkan Neptunus mengoptimalkan upserts di sekitar. IDs Peningkatan berbasis ID dapat secara signifikan lebih efisien daripada peningkatan berbasis properti dalam skenario modifikasi yang sangat bersamaan.

Bagian atas simpul berantai

Anda dapat menghubungkan bagian atas simpul bersama-sama untuk menyisipkannya dalam batch:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org'))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org'))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-3')
                         .property('email', 'person-3@example.org'))
 .id()
```

Tepi yang menjulang

Anda dapat menggunakan tepi IDs untuk menaikkan tepi dengan cara yang sama Anda menaikkan simpul menggunakan simpul kustom. IDs Sekali lagi, ini adalah pendekatan yang disukai karena memungkinkan Neptunus untuk mengoptimalkan kueri. Misalnya, kueri berikut membuat tepi berdasarkan ID tepinya jika belum ada, atau menggunakannya kembali jika ada. Kueri juga menggunakan simpul `from` dan `to` simpul jika perlu membuat tepi baru. IDs

```
g.E('e-1')
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2'))
                        .property(id, 'e-1'))
 .id()
```

Banyak aplikasi menggunakan simpul khusus IDs, tetapi meninggalkan Neptunus untuk menghasilkan tepi. IDs Jika Anda tidak tahu ID tepi, tetapi Anda tahu `from` dan `to` simpul IDs, Anda dapat menggunakan formulasi ini untuk meningkatkan tepi:

```
g.V('v-1')
```

```
 .outE('KNOWS')
 .where(inV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2')))
 .id()
```

Perhatikan bahwa langkah simpul dalam `where()` klausa harus `inV()` (atau `outV()` jika Anda pernah `inE()` menemukan tepi), bukan. `otherV()` Jangan gunakan`otherV()`, di sini, atau kueri tidak akan dioptimalkan dan kinerja akan menurun. Misalnya, Neptunus tidak akan mengoptimalkan kueri berikut:

```
// Unoptimized upsert, because of otherV()
g.V('v-1')
 .outE('KNOWS')
 .where(otherV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2')))
 .id()
```

Jika Anda tidak tahu tepi atau simpul IDs di depan, Anda dapat meningkatkan menggunakan properti simpul:

```
g.V()
 .hasLabel('Person')
 .has('name', 'person-1')
 .outE('LIVES_IN')
 .where(inV().hasLabel('City').has('name', 'city-1'))
 .fold()
 .coalesce(unfold(),
           addE('LIVES_IN').from(V().hasLabel('Person')
                                    .has('name', 'person-1'))
                           .to(V().hasLabel('City')
                                  .has('name', 'city-1')))
 .id()
```

Seperti halnya vertex upserts, lebih baik menggunakan edge upserts berbasis ID menggunakan ID tepi atau `from` dan `to` simpul, daripada upsert berbasis properti IDs, sehingga Neptunus dapat sepenuhnya mengoptimalkan upsert.

Memeriksa **`from`** dan **`to`** keberadaan simpul

Perhatikan konstruksi langkah-langkah yang membuat tepi baru:`addE().from().to()`. Konstruksi ini memastikan bahwa kueri memeriksa keberadaan simpul `from` dan `to` simpul. Jika salah satu dari ini tidak ada, kueri mengembalikan kesalahan sebagai berikut:

```
{
  "detailedMessage": "Encountered a traverser that does not map to a value for child...
  "code": "IllegalArgumentException",
  "requestId": "..."
}
```

Jika mungkin salah satu `from` atau `to` simpul tidak ada, Anda harus mencoba untuk meningkatkannya sebelum menaikkan tepi di antara keduanya. Lihat Menggabungkan vertex dan edge upserts.

Ada konstruksi alternatif untuk membuat tepi yang tidak boleh Anda gunakan:`V().addE().to()`. Itu hanya menambahkan tepi jika `from` simpul ada. Jika `to` simpul tidak ada, kueri menghasilkan kesalahan, seperti yang dijelaskan sebelumnya, tetapi jika `from` simpul tidak ada, itu diam-diam gagal memasukkan tepi, tanpa menghasilkan kesalahan apa pun. Misalnya, upsert berikut selesai tanpa menaikkan tepi jika `from` simpul tidak ada:

```
// Will not insert edge if from vertex does not exist
g.V('v-1')
 .outE('KNOWS')
 .where(inV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
           V('v-1').addE('KNOWS')
                   .to(V('v-2')))
 .id()
```

Bagian atas tepi rantai

Jika Anda ingin menghubungkan edge upserts bersama-sama untuk membuat permintaan batch, Anda harus memulai setiap upsert dengan pencarian simpul, bahkan jika Anda sudah tahu tepi. IDs

Jika Anda sudah tahu tepi IDs yang ingin Anda sertakan, dan simpul dan `to` simpul, Anda dapat menggunakan formulasi ini: IDs `from`

```
g.V('v-1')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
           V('v-1').addE('KNOWS')
                   .to(V('v-2'))
                   .property(id, 'e-1'))
 .V('v-3')
 .outE('KNOWS')
 .hasId('e-2').fold()
 .coalesce(unfold(),
           V('v-3').addE('KNOWS')
                   .to(V('v-4'))
                   .property(id, 'e-2'))
 .V('v-5')
 .outE('KNOWS')
 .hasId('e-3')
 .fold()
 .coalesce(unfold(),
           V('v-5').addE('KNOWS')
                   .to(V('v-6'))
                   .property(id, 'e-3'))
 .id()
```

Mungkin skenario peningkatan tepi batch yang paling umum adalah Anda mengetahui `from` dan `to` simpulnya IDs, tetapi tidak tahu tepi IDs yang ingin Anda tingkatkan. Dalam hal ini, gunakan formulasi berikut:

```
g.V('v-1')
 .outE('KNOWS')
 .where(inV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
           V('v-1').addE('KNOWS')
                   .to(V('v-2')))

 .V('v-3')
 .outE('KNOWS')
```

```
 .where(inV().hasId('v-4'))
 .fold()
 .coalesce(unfold(),
           V('v-3').addE('KNOWS')
                   .to(V('v-4')))
 .V('v-5')
 .outE('KNOWS')
 .where(inV().hasId('v-6'))
 .fold()
 .coalesce(unfold(),
           V('v-5').addE('KNOWS').to(V('v-6')))
 .id()
```

Jika Anda mengetahui IDs tepi yang ingin Anda sertakan, tetapi tidak tahu simpulnya `from` dan IDs `to` simpulnya (ini tidak biasa), Anda dapat menggunakan formulasi ini:

```
g.V()
 .hasLabel('Person')
 .has('email', 'person-1@example.org')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
           V().hasLabel('Person')
              .has('email', 'person-1@example.org')
              .addE('KNOWS')
              .to(V().hasLabel('Person')
                     .has('email', 'person-2@example.org'))
                     .property(id, 'e-1'))
 .V()
 .hasLabel('Person')
 .has('email', 'person-3@example.org')
 .outE('KNOWS')
 .hasId('e-2')
 .fold()
 .coalesce(unfold(),
           V().hasLabel('Person')
              .has('email', 'person-3@example.org')
              .addE('KNOWS')
              .to(V().hasLabel('Person')
                     .has('email', 'person-4@example.org'))
              .property(id, 'e-2'))
 .V()
```

```
 .hasLabel('Person')
 .has('email', 'person-5@example.org')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
           V().hasLabel('Person')
              .has('email', 'person-5@example.org')
              .addE('KNOWS')
              .to(V().hasLabel('Person')
                     .has('email', 'person-6@example.org'))
                     .property(id, 'e-3'))
 .id()
```

Menggabungkan vertex dan edge upserts

Terkadang Anda mungkin ingin meningkatkan kedua simpul dan tepi yang menghubungkannya. Anda dapat mencampur contoh batch yang disajikan di sini. Contoh berikut meningkatkan 3 simpul dan 2 tepi:

```
g.V('p-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-1')
                         .property('email', 'person-1@example.org'))
 .V('p-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-2')
                         .property('name', 'person-2@example.org'))
 .V('c-1')
 .fold()
 .coalesce(unfold(),
           addV('City').property(id, 'c-1')
                       .property('name', 'city-1'))
 .V('p-1')
 .outE('LIVES_IN')
 .where(inV().hasId('c-1'))
 .fold()
 .coalesce(unfold(),
           V('p-1').addE('LIVES_IN')
                   .to(V('c-1')))
 .V('p-2')
```

```
  .outE('LIVES_IN')
  .where(inV().hasId('c-1'))
  .fold()
  .coalesce(unfold(),
            V('p-2').addE('LIVES_IN')
                    .to(V('c-1')))
  .id()
```

Mencampur upserts dan sisipan

Terkadang Anda mungkin ingin meningkatkan kedua simpul dan tepi yang menghubungkannya. Anda dapat mencampur contoh batch yang disajikan di sini. Contoh berikut meningkatkan 3 simpul dan 2 tepi:

Upserts biasanya melanjutkan satu elemen pada satu waktu. Jika Anda tetap berpegang pada pola upsert yang disajikan di sini, setiap operasi upsert memancarkan satu traverser, yang menyebabkan operasi berikutnya dijalankan hanya sekali.

Namun, terkadang Anda mungkin ingin mencampur upserts dengan sisipan. Ini bisa terjadi, misalnya, jika Anda menggunakan tepi untuk mewakili contoh tindakan atau peristiwa. Permintaan mungkin menggunakan upserts untuk memastikan bahwa semua simpul yang diperlukan ada, dan kemudian menggunakan sisipan untuk menambahkan tepi. Dengan permintaan semacam ini, perhatikan potensi jumlah pelintas yang dipancarkan dari setiap operasi.

Perhatikan contoh berikut, yang mencampur upsert dan sisipan untuk menambahkan tepi yang mewakili peristiwa ke dalam grafik:

```
// Fully optimized, but inserts too many edges
g.V('p-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-1')
                         .property('email', 'person-1@example.org'))
 .V('p-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-2')
                         .property('name', 'person-2@example.org'))
 .V('p-3')
 .fold()
 .coalesce(unfold(),
```

```
          addV('Person').property(id, 'p-3')
                        .property('name', 'person-3@example.org'))
 .V('c-1')
 .fold()
 .coalesce(unfold(),
          addV('City').property(id, 'c-1')
                      .property('name', 'city-1'))
 .V('p-1', 'p-2')
 .addE('FOLLOWED')
 .to(V('p-1'))
 .V('p-1', 'p-2', 'p-3')
 .addE('VISITED')
 .to(V('c-1'))
 .id()
```

Kueri harus menyisipkan 5 tepi: 2 tepi DIIKUTI dan 3 tepi VISITED. Namun, kueri sebagai tertulis menyisipkan 8 tepi: 2 DIIKUTI dan 6 DIKUNJUNGI. Alasan untuk ini adalah bahwa operasi yang menyisipkan 2 tepi FOLLOW memancarkan 2 traversers, menyebabkan operasi penyisipan berikutnya, yang menyisipkan 3 tepi, dieksekusi dua kali.

Perbaikannya adalah menambahkan `fold()` langkah setelah setiap operasi yang berpotensi memancarkan lebih dari satu traverser:

```
g.V('p-1')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'p-1')
                        .property('email', 'person-1@example.org'))
 .V('p-2')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'p-2').
                        .property('name', 'person-2@example.org'))
 .V('p-3')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'p-3').
                        .property('name', 'person-3@example.org'))
 .V('c-1')
 .fold().
 .coalesce(unfold(),
           addV('City').property(id, 'c-1').
                       .property('name', 'city-1'))
```

```
  .V('p-1', 'p-2')
  .addE('FOLLOWED')
  .to(V('p-1'))
  .fold()
  .V('p-1', 'p-2', 'p-3')
  .addE('VISITED')
  .to(V('c-1')).
  .id()
```

Di sini kita telah memasukkan `fold()` langkah setelah operasi yang menyisipkan tepi DIIKUTI. Ini menghasilkan traverser tunggal, yang kemudian menyebabkan operasi berikutnya dieksekusi hanya sekali.

Kelemahan dari pendekatan ini adalah bahwa query sekarang tidak sepenuhnya dioptimalkan, karena tidak `fold()` dioptimalkan. Operasi insert yang mengikuti sekarang tidak `fold()` akan dioptimalkan.

Jika Anda perlu menggunakan `fold()` untuk mengurangi jumlah pelintas atas nama langkah-langkah selanjutnya, cobalah untuk memesan operasi Anda sehingga yang paling murah menempati bagian kueri yang tidak dioptimalkan.

Upserts yang memodifikasi simpul dan tepi yang ada

Terkadang Anda ingin membuat simpul atau tepi jika tidak ada, dan kemudian menambahkan atau memperbarui properti ke sana, terlepas dari apakah itu simpul atau tepi baru atau yang sudah ada.

Untuk menambah atau memodifikasi properti, gunakan `property()` langkah. Gunakan langkah ini di luar `coalesce()` langkah. Jika Anda mencoba memodifikasi properti simpul atau tepi yang ada di dalam `coalesce()` langkah, kueri mungkin tidak dioptimalkan oleh mesin kueri Neptunus.

Kueri berikut menambahkan atau memperbarui properti penghitung pada setiap simpul yang diupsert. Setiap `property()` langkah memiliki kardinalitas tunggal untuk memastikan bahwa nilai baru menggantikan nilai yang ada, daripada ditambahkan ke serangkaian nilai yang ada.

```
g.V('v-1')
  .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-1')
                          .property('email', 'person-1@example.org'))
  .property(single, 'counter', 1)
  .V('v-2')
  .fold()
```

```
  .coalesce(unfold(),
            addV('Person').property(id, 'v-2')
                          .property('email', 'person-2@example.org'))
  .property(single, 'counter', 2)
  .V('v-3')
  .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-3')
                          .property('email', 'person-3@example.org'))
  .property(single, 'counter', 3)
  .id()
```

Jika Anda memiliki nilai properti, seperti nilai `lastUpdated` stempel waktu, yang berlaku untuk semua elemen yang muncul, Anda dapat menambahkan atau memperbaruinya di akhir kueri:

```
g.V('v-1')
  .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-1')
                          .property('email', 'person-1@example.org'))
  .V('v-2').
  .fold().
  .coalesce(unfold(),
            addV('Person').property(id, 'v-2')
                          .property('email', 'person-2@example.org'))
  .V('v-3')
  .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-3')
                          .property('email', 'person-3@example.org'))
  .V('v-1', 'v-2', 'v-3')
  .property(single, 'lastUpdated', datetime('2020-02-08'))
  .id()
```

Jika ada kondisi tambahan yang menentukan apakah simpul atau tepi harus dimodifikasi lebih lanjut atau tidak, Anda dapat menggunakan `has()` langkah untuk memfilter elemen yang akan diterapkan modifikasi. Contoh berikut menggunakan `has()` langkah untuk memfilter simpul naik berdasarkan nilai properti mereka. `version` Kueri kemudian memperbarui ke 3 `version` dari setiap simpul yang `version` kurang dari 3:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org')
                         .property('version', 3))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org')
                         .property('version', 3))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-3')
                         .property('email', 'person-3@example.org')
                         .property('version', 3))
 .V('v-1', 'v-2', 'v-3')
 .has('version', lt(3))
 .property(single, 'version', 3)
 .id()
```

Menganalisis eksekusi kueri Neptune menggunakan **`explain`** Gremlin

Amazon Neptune telah menambahkan fitur Gremlin bernama explain. Fitur ini adalah alat swalayan untuk memahami pendekatan eksekusi yang diambil oleh mesin Neptune. Anda memintanya dengan menambahkan parameter `explain` ke panggilan HTTP yang mengirimkan kueri Gremlin.

Fitur `explain` menyediakan informasi tentang struktur logis rencana eksekusi kueri. Anda dapat menggunakan informasi ini untuk mengidentifikasi potensi evaluasi dan hambatan eksekusi dan menyetel kueri Anda, seperti yang dijelaskan dalam Menyetel kueri Gremlin. Anda juga dapat menggunakan petunjuk kueri untuk meningkatkan rencana eksekusi kueri.

Topik

- Memahami bagaimana kueri Gremlin bekerja di Neptune
- Menggunakan API explain Gremlin di Neptune
- API profile Gremlin di Neptune
- Menyetel kueri Gremlin menggunakan explain dan profile
- Dukungan langkah Gremlin asli di Amazon Neptune

Memahami bagaimana kueri Gremlin bekerja di Neptune

Untuk mengambil keuntungan penuh dari laporan `explain` dan `profile` Gremlin di Amazon Neptune, akan sangat membantu untuk memahami beberapa informasi latar belakang tentang kueri Gremlin.

Topik

- Pernyataan Gremlin di Neptune
- Bagaimana Neptune memproses kueri Gremlin menggunakan indeks pernyataan
- Bagaimana kueri Gremlin diproses di Neptune

Pernyataan Gremlin di Neptune

Data grafik properti di Amazon Neptune terdiri dari pernyataan empat-posisi (quad). Masing-masing pernyataan ini merupakan unit atom individu data grafik properti. Untuk informasi selengkapnya, lihat Model Data Grafik Neptune. Mirip dengan model data Resource Description Framework (RDF), empat posisi ini adalah sebagai berikut:

- `subject (S)`
- `predicate (P)`
- `object (O)`
- `graph (G)`

Setiap pernyataan adalah penegasan tentang satu sumber daya atau lebih. Misalnya, sebuah pernyataan dapat menegaskan adanya hubungan antara dua sumber daya, atau dapat melampirkan properti (pasangan nilai-kunci) ke beberapa sumber daya.

Anda dapat menganggap predikat sebagai kata kerja pernyataan, menggambarkan jenis hubungan atau properti. Objeknya adalah target hubungan, atau nilai properti. Posisi grafik adalah opsional dan dapat digunakan dalam berbagai cara. Untuk data grafik properti (PG) Neptune, bisa tidak digunakan (grafik null) atau digunakan untuk mewakili pengidentifikasi untuk edge. Satu set pernyataan dengan pengidentifikasi sumber daya bersama menciptakan grafik.

Ada tiga kelas pernyataan dalam model data grafik properti Neptune:

Topik

- Pernyataan Label Vertex Gremlin

- Pernyataan Gremlin Edge
- Pernyataan Properti Gremlin

Pernyataan Label Vertex Gremlin

Pernyataan label Vertex di Neptune melayani dua tujuan:

- Mereka melacak label untuk vertex.
- Kehadiran setidaknya satu dari pernyataan ini adalah yang menyiratkan adanya vertex tertentu dalam grafik.

Subjek dari pernyataan ini adalah pengidentifikasi vertex, dan objeknya adalah label, yang keduanya ditentukan oleh pengguna. Anda menggunakan predikat tetap khusus untuk pernyataan ini, ditampilkan sebagai `<~label>`, dan pengidentifikasi grafik default (grafik null), ditampilkan sebagai `<~>`.

Contohnya, pertimbangkan traversal `addV` berikut ini.

```
g.addV("Person").property(id, "v1")
```

Traversal ini menghasilkan pernyataan berikut yang ditambahkan ke grafik.

```
StatementEvent[Added(<v1> <~label> <Person> <~>) .]
```

Pernyataan Gremlin Edge

Sebuah pernyataan edge Gremlin adalah yang menyiratkan adanya edge antara dua vertex dalam grafik di Neptune. Subjek (S) dari pernyataan edge adalah vertex `from` sumber. Predikat (P) adalah label edge yang disediakan pengguna. Objek (O) adalah vertex `to` target. Grafik (G) adalah pengenal edge yang disediakan pengguna.

Contohnya, pertimbangkan traversal `addE` berikut ini.

```
g.addE("knows").from(V("v1")).to(V("v2")).property(id, "e1")
```

Traversal menghasilkan pernyataan berikut yang ditambahkan ke grafik.

```
StatementEvent[Added(<v1> <knows> <v2> <e1>) .]
```

Pernyataan Properti Gremlin

Sebuah pernyataan properti Gremlin di Neptune menegaskan nilai properti individu untuk vertex atau edge. Subjek adalah pengidentifikasi vertex atau edge yang disediakan pengguna. Predikat adalah nama properti (kunci), dan objek adalah nilai properti individu. Grafik (G) adalah sekali lagi pengidentifikasi grafik default, grafik null, ditampilkan sebagai <~>.

Perhatikan contoh properti simpul berikut.

```
g.V("v1").property("name", "John")
```

Pernyataan ini menghasilkan berikut ini.

```
StatementEvent[Added(<v1> <name> "John" <~>) .]
```

Pernyataan properti berbeda dari yang lain dalam hal bahwa objek mereka adalah nilai primitif (`string`, `date`, `byte`, `short`, `int`, `long`, `float`, atau `double`). Objek mereka bukan pengidentifikasi sumber daya yang dapat digunakan sebagai subjek penegasan lain.

Untuk multi-properti, setiap nilai properti individu dalam set menerima pernyataannya sendiri.

```
g.V("v1").property(set, "phone", "956-424-2563").property(set, "phone", "956-354-3692
 (tel:9563543692)")
```

Hal ini menyebabkan hal berikut ini.

```
StatementEvent[Added(<v1> <phone> "956-424-2563" <~>) .]
StatementEvent[Added(<v1> <phone> "956-354-3692" <~>) .]
```

Properti tepi ditangani mirip dengan properti simpul, tetapi gunakan pengenal tepi di posisi (S). Misalnya, menambahkan properti ke tepi:

```
g.E("e1").property("weight", 0.8)
```

Ini menghasilkan pernyataan berikut yang ditambahkan ke grafik.

```
StatementEvent[Added(<e1> <weight> 0.8 <~>) .]
```

Bagaimana Neptune memproses kueri Gremlin menggunakan indeks pernyataan

Pernyataam diakses di Amazon Neptune dengan cara tiga indeks pernyataan, seperti yang dijelaskan dalam Bagaimana Pernyataan Diindeks di Neptune. Neptune mengekstrak sebuah pola pernyataan dari kueri Gremlin di mana beberapa posisi diketahui, dan sisanya dibiarkan untuk penemuan oleh pencarian indeks.

Neptune mengasumsikan bahwa ukuran skema grafik properti tidak besar. Ini berarti bahwa jumlah label edge yang berbeda dan nama properti cukup rendah, sehingga jumlah predikat yang berbeda rendah. Neptune melacak predikat yang berbeda dalam indeks terpisah. Menggunakan cache predikat ini untuk melakukan pemindaian serikat `{ all P x POGS }` daripada menggunakan indeks OSGP. Menghindari kebutuhan untuk indeks OSGP traversal terbalik menghemat ruang penyimpanan dan throughput beban.

Neptunus Explain/Profile Gremlin API memungkinkan Anda mendapatkan jumlah predikat dalam grafik Anda. Anda kemudian dapat menentukan apakah aplikasi Anda membatalkan asumsi Neptune bahwa skema grafik properti Anda kecil.

Contoh berikut membantu menggambarkan bagaimana Neptune menggunakan indeks untuk memproses kueri Gremlin.

Pertanyaan: Apa label simpul? **v1**

```
Gremlin code:      g.V('v1').label()
Pattern:           (<v1>, <~label>, ?, ?)
Known positions:   SP
Lookup positions:  OG
Index:             SPOG
Key range:         <v1>:<~label>:*
```

Pertanyaan: Apa tepi luar 'tahu' dari simpul? **v1**

```
Gremlin code:      g.V('v1').out('knows')
Pattern:           (<v1>, <knows>, ?, ?)
Known positions:   SP
Lookup positions:  OG
Index:             SPOG
Key range:         <v1>:<knows>:*
```

Pertanyaan: Simpul mana yang memiliki label **Person** simpul?

```
Gremlin code:      g.V().hasLabel('Person')
Pattern:           (?, <~label>, <Person>, <~>)
Known positions:   POG
Lookup positions:  S
Index:             POGS
Key range:         <~label>:<Person>:<~>:*
```

Pertanyaan: Apa from/to simpul dari tepi yang diberikan? **e1**

```
Gremlin code:      g.E('e1').bothV()
Pattern:           (?, ?, ?, <e1>)
Known positions:   G
Lookup positions:  SPO
Index:             GPSO
Key range:         <e1>:*
```

Satu indeks pernyataan yang tidak Neptune miliki adalah indeks OSGP traversal terbalik. Indeks ini dapat digunakan untuk mengumpulkan semua edge masuk di semua label edge, seperti dalam contoh berikut.

Pertanyaan: Apa **v1** simpul yang berdekatan yang masuk?

```
Gremlin code:      g.V('v1').in()
Pattern:           (?, ?, <v1>, ?)
Known positions:   O
Lookup positions:  SPG
Index:             OSGP  // <-- Index does not exist
```

Bagaimana kueri Gremlin diproses di Neptune

Di Amazon Neptune, traversal yang lebih kompleks dapat diwakili oleh serangkaian pola yang menciptakan relasi berdasarkan definisi variabel bernama yang dapat dibagi di seluruh pola untuk membuat gabungan. Seperti yang ditunjukkan dalam contoh berikut.

Pertanyaan: Apa lingkungan dua hop dari vertex? **v1**

```
Gremlin code:      g.V(‘v1’).out('knows').out('knows').path()
Pattern:           (?1=<v1>, <knows>, ?2, ?) X Pattern(?2, <knows>, ?3, ?)

The pattern produces a three-column relation (?1, ?2, ?3) like this:
                   ?1     ?2     ?3
```

```
================
v1      v2      v3
v1      v2      v4
v1      v5      v6
```

Dengan berbagi variabel ?2 di dua pola (pada posisi O dalam pola pertama dan posisi S dari pola kedua), Anda membuat gabungan dari tetangga hop pertama ke tetangga hop kedua. Setiap solusi Neptunus memiliki binding untuk tiga variabel bernama, yang dapat digunakan untuk membuat ulang TinkerPop Traverser (termasuk informasi jalur).

Langkah pertama dalam pemrosesan kueri Gremlin adalah mengurai kueri menjadi objek TinkerPop Traversal, yang terdiri dari serangkaian langkah. TinkerPop Langkah-langkah ini, yang merupakan bagian dari TinkerPop proyek Apache open-source, adalah operator logis dan fisik yang menyusun traversal Gremlin dalam implementasi referensi. Keduanya digunakan untuk mewakili model kueri. Keduanya adalah operator yang dapat dieksekusi yang dapat menghasilkan solusi sesuai dengan semantik operator yang mereka wakili. Misalnya, keduanya `.V()` diwakili dan dieksekusi oleh TinkerPop GraphStep.

Karena off-the-shelf TinkerPop langkah-langkah ini dapat dieksekusi, TinkerPop Traversal semacam itu dapat mengeksekusi kueri Gremlin apa pun dan menghasilkan jawaban yang benar. Namun, ketika dieksekusi terhadap grafik besar, TinkerPop langkah-langkah terkadang bisa sangat tidak efisien dan lambat. Alih-alih menggunakannya, Neptune mencoba mengubah traversal menjadi bentuk deklaratif yang terdiri dari grup pola, seperti yang dijelaskan sebelumnya.

Neptune saat ini tidak mendukung semua operator (langkah) Gremlin di mesin kueri aslinya. Jadi Neptune mencoba melakukan collaps pada sebanyak mungkin langkah ke dalam satu `NeptuneGraphQueryStep`, yang berisi rencana kueri logis deklaratif untuk semua langkah-langkah yang telah dikonversi. Idealnya, semua langkah dikonversi. Tetapi ketika sebuah langkah ditemukan yang tidak dapat dikonversi, Neptunus keluar dari eksekusi asli dan menunda semua eksekusi kueri dari titik itu ke langkah-langkahnya. TinkerPop Neptune tidak mencoba untuk menjalin masuk dan keluar dari eksekusi asli.

Setelah langkah-langkah diterjemahkan ke dalam rencana kueri logis, Neptune menjalankan serangkaian pengoptimal kueri yang menulis ulang rencana kueri berdasarkan analisis statis dan perkiraan kardinalitas. Pengoptimal ini melakukan hal-hal seperti mengatur ulang operator berdasarkan jumlah rentang, memangkas operator yang tidak perlu atau berlebihan, mengatur ulang filter, mendorong operator ke dalam kelompok yang berbeda, dan sebagainya.

Setelah rencana kueri yang dioptimalkan diproduksi, Neptune menciptakan alur operator fisik yang melakukan pekerjaan mengeksekusi kueri. Ini termasuk membaca data dari indeks pernyataan, melakukan penggabungan berbagai jenis, penyaringan, pengurutan, dan sebagainya. Pipeline menghasilkan aliran solusi yang kemudian diubah kembali menjadi aliran objek TinkerPop Traverser.

Serialisasi hasil kueri

Amazon Neptunus saat ini mengandalkan TinkerPop serializer pesan respons untuk mengonversi hasil kueri TinkerPop (Traversers) menjadi data serial untuk dikirim melalui kabel kembali ke klien. Format serialisasi ini cenderung cukup verbose.

Sebagai contoh, untuk melakukan serialisasi hasil dari kueri vertex seperti `g.V().limit(1)`, mesin kueri Neptune harus melakukan pencarian tunggal untuk memproduksi hasil kueri. Namun, serializer `GraphSON` akan melakukan sejumlah besar pencarian tambahan untuk mengemas vertex ke dalam format serialisasi. Serializer harus melakukan satu pencarian untuk mendapatkan label, satu pencarian untuk mendapatkan kunci properti, dan satu pencarian per kunci properti untuk vertex untuk mendapatkan semua nilai untuk setiap kunci.

Beberapa format serialisasi lebih efisien, tetapi semua memerlukan pencarian tambahan. Selain itu, TinkerPop serializer tidak mencoba menghindari penelusuran duplikat, seringkali mengakibatkan banyak pencarian diulang secara tidak perlu.

Hal ini membuat sangat penting untuk menulis kueri Anda sehingga mereka meminta secara khusus hanya untuk informasi yang mereka butuhkan. Misalnya, `g.V().limit(1).id()` akan mengembalikan hanya ID vertex dan menghilangkan semua pencarian serializer tambahan. API `profile` Gremlin di Neptune memungkinkan Anda untuk melihat berapa banyak panggilan pencarian yang dilakukan selama eksekusi kueri dan selama serialisasi.

Menggunakan API **`explain`** Gremlin di Neptune

API `explain` Gremlin Amazon Neptune mengembalikan rencana kueri yang akan dieksekusi jika kueri tertentu dijalankan. Karena API tidak benar-benar menjalankan kueri, rencana dikembalikan hampir seketika.

Ini berbeda dari langkah TinkerPop .explain () sehingga dapat melaporkan informasi khusus ke mesin Neptunus.

Informasi yang terkandung dalam laporan **`explain`** Gremlin

Laporan `explain` berisi informasi berikut:

- String kueri seperti yang diminta.
- Traversal asli. Ini adalah objek TinkerPop Traversal yang dihasilkan dengan mengurai string kueri menjadi beberapa TinkerPop langkah. Ini setara dengan kueri asli yang dihasilkan dengan menjalankan `.explain()` kueri terhadap kueri TinkerPop TinkerGraph.
- Traversal yang dikonversi. Ini adalah Traversal Neptunus yang dihasilkan dengan mengubah TinkerPop Traversal menjadi representasi rencana kueri logis Neptunus. Dalam banyak kasus, seluruh TinkerPop traversal diubah menjadi dua langkah Neptunus: satu yang mengeksekusi seluruh query `NeptuneGraphQueryStep` () dan satu yang mengubah output mesin kueri Neptunus kembali menjadi Traversers (). TinkerPop `NeptuneTraverserConverterStep`
- Traversal yang dioptimalkan. Ini adalah versi yang dioptimalkan dari rencana kueri Neptune setelah dijalankan melalui serangkaian pengoptimalan mengurangi kerja statis yang menulis ulang kueri berdasarkan analisis statis dan perkiraan kardinalitas. Pengoptimal ini melakukan hal-hal seperti mengatur ulang operator berdasarkan jumlah rentang, memangkas operator yang tidak perlu atau berlebihan, mengatur ulang filter, mendorong operator ke dalam kelompok yang berbeda, dan sebagainya.
- Hitungan predikat. Karena strategi pengindeksan Neptune yang dijelaskan sebelumnya, memiliki sejumlah besar predikat yang berbeda dapat menyebabkan masalah performa. Hal ini terutama berlaku untuk kueri yang menggunakan operator traversal terbalik tanpa label edge (`.in` atau `.both`). Jika operator tersebut digunakan dan jumlah predikat cukup tinggi, `explain` menampilkan pesan peringatan.
- Informasi DFE. Ketika mesin alternatif DFE diaktifkan, komponen traversal berikut mungkin muncul dalam traversal yang dioptimalkan:
  - **`DFEStep`**— Langkah DFE Neptunus yang dioptimalkan dalam traversal yang berisi anak. `DFENode` `DFEStep`merupakan bagian dari rencana kueri yang dijalankan di mesin DFE.
  - **`DFENode`**— Berisi representasi menengah sebagai satu atau lebih anak`DFEJoinGroupNodes`.
  - **`DFEJoinGroupNode`**— Merupakan gabungan dari satu atau lebih `DFENode` atau `DFEJoinGroupNode` elemen.
  - **`NeptuneInterleavingStep`**— Langkah DFE Neptunus yang dioptimalkan dalam traversal yang berisi anak. `DFEStep`

    Juga berisi `stepInfo` elemen yang berisi informasi tentang traversal, seperti elemen perbatasan, elemen jalur yang digunakan, dan sebagainya. Informasi ini digunakan untuk memproses anak`DFEStep`.

Cara mudah untuk mengetahui apakah permintaan Anda sedang dievaluasi oleh DFE adalah memeriksa apakah output `explain` berisi `DFEStep`. Setiap bagian dari traversal yang bukan bagian dari tidak `DFEStep` akan dieksekusi oleh DFE dan akan dieksekusi oleh mesin. TinkerPop

Lihat Contoh dengan DFE diaktifkan untuk laporan contoh.

Sintaks **`explain`** Gremlin

Sintaks API `explain` sama seperti untuk API HTTP untuk kueri, kecuali bahwa ia menggunakan `/gremlin/explain` sebagai titik akhir, bukan `/gremlin`, seperti pada contoh berikut.

```
curl -X POST https://your-neptune-endpoint:port/gremlin/explain -d
 '{"gremlin":"g.V().limit(1)"}'
```

Kueri sebelumnya akan menghasilkan output berikut.

```
********************************************************
                Neptune Gremlin Explain
********************************************************

Query String
============
g.V().limit(1)

Original Traversal
==================
[GraphStep(vertex,[]), RangeGlobalStep(0,1)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
        }, finishers=[limit(1)], annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
```

```
==================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, finishers=[limit(1)], annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 18
```

Langkah Belum Dikonversi TinkerPop

Idealnya, semua TinkerPop langkah dalam traversal memiliki cakupan operator Neptunus asli. Ketika ini tidak terjadi, Neptunus kembali TinkerPop pada eksekusi langkah untuk kesenjangan dalam jangkauan operatornya. Jika traversal menggunakan langkah yang belum Neptune miliki cakupan aslinya, laporan `explain` menampilkan peringatan yang menunjukkan dimana kesenjangan terjadi.

Ketika sebuah langkah tanpa operator Neptunus asli yang sesuai ditemukan, seluruh traversal dari titik itu ke depan dijalankan TinkerPop menggunakan langkah-langkah, bahkan jika langkah selanjutnya memang memiliki operator Neptunus asli.

Pengecualian untuk ini adalah ketika pencarian teks lengkap Neptune dipanggil. NeptuneSearchStep Mengimplementasikan langkah-langkah tanpa padanan asli sebagai langkah pencarian teks lengkap.

Contoh output **`explain`** di mana semua langkah dalam kueri memiliki ekuivalen asli

Berikut ini adalah contoh laporan `explain` untuk kueri di mana semua langkah memiliki ekuivalen asli:

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============
g.V().out()
```

```
Original Traversal
==================
[GraphStep(vertex,[]), VertexStep(OUT,vertex)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 18
```

Contoh di mana beberapa langkah dalam kueri tidak memiliki ekuivalen asli

Neptune menangani `GraphStep` dan `VertexStep` secara native, tetapi jika Anda memperkenalkan `FoldStep` dan `UnfoldStep`, output `explain` yang dihasilkan berbeda:

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************
```

```
Query String
============
g.V().fold().unfold().out()

Original Traversal
==================
[GraphStep(vertex,[]), FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep,
    NeptuneMemoryTrackerStep
]
+ not converted into Neptune steps: [FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

WARNING: >> FoldStep << is not supported natively yet
```

Dalam hal ini, `FoldStep` menghentikan Anda dari eksekusi asli. Namun bahkan `VertexStep` berikutnya tidak lagi ditangani secara native karena muncul hilir dari langkah `Fold/Unfold`.

Untuk kinerja dan penghematan biaya, penting bagi Anda untuk mencoba merumuskan traversal sehingga jumlah maksimum pekerjaan yang mungkin dilakukan secara asli di dalam mesin kueri Neptunus, bukan dengan implementasi langkah. TinkerPop

Contoh kueri yang menggunakan Neptunus full-text-search

Kueri berikut menggunakan pencarian teks lengkap Neptune:

```
g.withSideEffect("Neptune#fts.endpoint", "some_endpoint")
  .V()
  .tail(100)
  .has("Neptune#fts mark*")
  -------
  .has("name", "Neptune#fts mark*")
  .has("Person", "name", "Neptune#fts mark*")
```

Bagian `.has("name", "Neptune#fts mark*")` membatasi pencarian ke vertex dengan `name`, sementara `.has("Person", "name", "Neptune#fts mark*")` membatasi pencarian ke vertex dengan `name` dan label `Person`. Hal ini menghasilkan traversal berikut di laporan `explain`:

```
Final Traversal
[NeptuneGraphQueryStep(Vertex) {
    JoinGroupNode {
        PatternNode[(?1, termid(1,URI), ?2, termid(0,URI)) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
    }, annotations={path=[Vertex(?1):GraphStep], maxVarId=4}
}, NeptuneTraverserConverterStep, NeptuneTailGlobalStep(10),
 NeptuneTinkerpopTraverserConverterStep, NeptuneSearchStep {
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=some_endpoint}
    }
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=some_endpoint}
    }
}]
```

Contoh penggunaan **`explain`** saat DFE diaktifkan

Berikut ini adalah contoh laporan `explain` ketika mesin kueri alternatif DFE diaktifkan:

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============

g.V().as("a").out().has("name", "josh").out().in().where(eq("a"))


Original Traversal
==================
[GraphStep(vertex,[])@[a], VertexStep(OUT,vertex), HasStep([name.eq(josh)]),
 VertexStep(OUT,vertex), VertexStep(IN,vertex), WherePredicateStep(eq(a))]

Converted Traversal
===================
Neptune steps:
[
    DFEStep(Vertex) {
      DFENode {
        DFEJoinGroupNode[ children={
          DFEPatternNode[(?1, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, ?2,
 <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>) . project DISTINCT[?1]
 {rangeCountEstimate=unknown}],
          DFEPatternNode[(?1, ?3, ?4, ?5) . project ALL[?1, ?4] graphFilters=(!
= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}]
        }, {rangeCountEstimate=unknown}
        ]
      } [Vertex(?1):GraphStep@[a], Vertex(?4):VertexStep]
    } ,
    NeptuneTraverserConverterDFEStep
]
+ not converted into Neptune steps: HasStep([name.eq(josh)]),
Neptune steps:
[
    NeptuneInterleavingStep {
      StepInfo[joinVars=[?7, ?1], frontierElement=Vertex(?7):HasStep,
 pathElements={a=(last,Vertex(?1):GraphStep@[a])}, listPathElement={}, indexTime=0ms],
      DFEStep(Vertex) {
        DFENode {
          DFEJoinGroupNode[ children={
```

```
            DFEPatternNode[(?7, ?8, ?9, ?10) . project ALL[?7, ?9]
 graphFilters=(!= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}],
            DFEPatternNode[(?12, ?11, ?9, ?13) . project ALL[?9, ?12]
 graphFilters=(!= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}]
          }, {rangeCountEstimate=unknown}
          ]
        } [Vertex(?9):VertexStep, Vertex(?12):VertexStep]
      }
    }
]
+ not converted into Neptune steps: WherePredicateStep(eq(a)),
Neptune steps:
[
    DFECleanupStep
]


Optimized Traversal
===================
Neptune steps:
[
    DFEStep(Vertex) {
      DFENode {
        DFEJoinGroupNode[ children={
          DFEPatternNode[(?1, ?3, ?4, ?5) . project ALL[?1, ?4] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=92233720368547758O7}]
        }, {rangeCountEstimate=unknown}
        ]
      } [Vertex(?1):GraphStep@[a], Vertex(?4):VertexStep]
    } ,
    NeptuneTraverserConverterDFEStep
]
+ not converted into Neptune steps: NeptuneHasStep([name.eq(josh)]),
Neptune steps:
[
    NeptuneMemoryTrackerStep,
    NeptuneInterleavingStep {
      StepInfo[joinVars=[?7, ?1], frontierElement=Vertex(?7):HasStep,
 pathElements={a=(last,Vertex(?1):GraphStep@[a])}, listPathElement={}, indexTime=0ms],
      DFEStep(Vertex) {
        DFENode {
          DFEJoinGroupNode[ children={
```

```
            DFEPatternNode[(?7, ?8, ?9, ?10) . project ALL[?7, ?9] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807}],
            DFEPatternNode[(?12, ?11, ?9, ?13) . project ALL[?9, ?12] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807}]
          }, {rangeCountEstimate=unknown}
          ]
        } [Vertex(?9):VertexStep, Vertex(?12):VertexStep]
      }
    }
]
+ not converted into Neptune steps: WherePredicateStep(eq(a)),
Neptune steps:
[
    DFECleanupStep
]


WARNING: >> [NeptuneHasStep([name.eq(josh)]), WherePredicateStep(eq(a))] << (or one of
 the children for each step) is not supported natively yet

Predicates
==========
# of predicates: 8
```

Lihat Informasi di `explain` untuk deskripsi bagian spesifik DFE dalam laporan.

API **`profile`** Gremlin di Neptune

API `profile` Gremlin Neptune menjalankan traversal Gremlin, mengumpulkan berbagai metrik tentang proses, dan menghasilkan laporan profil sebagai output.

Ini berbeda dari TinkerPop langkah.profile () sehingga dapat melaporkan informasi khusus untuk mesin Neptunus.

Laporan profil mencakup informasi berikut tentang rencana kueri:

- Alur operator fisik
- Operasi indeks untuk eksekusi dan serialisasi kueri
- Ukuran hasilnya

API `profile` menggunakan versi diperpanjang dari sintaks HTTP API untuk kueri, dengan `/gremlin/profile` sebagai titik akhir alih-alih `/gremlin`.

Parameter khusus untuk **`profile`** Gremlin Neptune

- profile.results — `boolean`, nilai yang diizinkan: `TRUE` dan `FALSE`, nilai default: `TRUE`.

  Jika true, hasil kueri dikumpulkan dan ditampilkan sebagai bagian dari laporan `profile`. Jika false, hanya jumlah hasil yang ditampilkan.
- profile.chop — `int`, nilai default: 250.

  Jika non-nol, menyebabkan hasil string akan dipotong pada jumlah karakter tersebut. Hal ini tidak menjaga semua hasil dari ditangkap. Ini hanya membatasi ukuran string dalam laporan profil. Jika diatur ke nol, string berisi semua hasil.
- profile.serializer – `string`, nilai default: `<null>`.

  Jika non-null, hasil yang dikumpulkan dikembalikan dalam pesan respons serial dalam format yang ditentukan oleh parameter ini. Jumlah operasi indeks yang diperlukan untuk menghasilkan pesan respons dilaporkan bersama dengan ukuran dalam byte yang akan dikirim ke klien.

  Nilai yang diizinkan adalah `<null>` atau salah satu jenis MIME yang valid atau nilai enum “Serializers” TinkerPop driver.

  ```
  "application/json" or "GRAPHSON"
  "application/vnd.gremlin-v1.0+json" or "GRAPHSON_V1"
  "application/vnd.gremlin-v1.0+json;types=false" or "GRAPHSON_V1_UNTYPED"
  "application/vnd.gremlin-v2.0+json" or "GRAPHSON_V2"
  "application/vnd.gremlin-v2.0+json;types=false" or "GRAPHSON_V2_UNTYPED"
  "application/vnd.gremlin-v3.0+json" or "GRAPHSON_V3"
  "application/vnd.gremlin-v3.0+json;types=false" or "GRAPHSON_V3_UNTYPED"
  "application/vnd.graphbinary-v1.0" or "GRAPHBINARY_V1"
  ```
- profile.indexOps — `boolean`, nilai yang diizinkan: `TRUE` dan `FALSE`, nilai default: `FALSE`.

  Jika true, menunjukkan laporan detail dari semua operasi indeks yang terjadi selama eksekusi dan serialisasi kueri. Peringatan: Laporan ini bisa verbose.

Contoh output dari **`profile`** Gremlin Neptune

Berikut ini adalah sampel kueri `profile`.

```
curl -k -X POST https://your-neptune-endpoint:port/gremlin/profile \
```

```
    -d '{"gremlin":"g.V().hasLabel(\"airport\").has(\"code\",
 \"AUS\").emit().repeat(in().simplePath()).times(2).limit(100)",
 "profile.serializer":"application/vnd.gremlin-v3.0+json"}'
```

Query ini menghasilkan laporan `profile` berikut ketika dieksekusi pada grafik sampel rute udara dari posting blog, Mari Saya Grafikkan Untuk Anda - Bagian 1 - Rute Udara.

```
*******************************************************
                Neptune Gremlin Profile
*******************************************************

Query String
==================
g.V().hasLabel("airport").has("code",
 "AUS").emit().repeat(in().simplePath()).times(2).limit(100)

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([~label.eq(airport), code.eq(AUS)]),
 RepeatStep(emit(true),[VertexStep(IN,vertex), PathFilterStep(simple),
 RepeatEndStep],until(loops(2))), RangeGlobalStep(0,100)]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "AUS", ?) . project ?1 .],
 {estimatedCardinality=1, indexTime=84, hashJoin=true, joinTime=3, actualTotalOutput=1}
            PatternNode[(?1, <~label>, ?2=<airport>, <~>) . project ask .],
 {estimatedCardinality=3374, indexTime=29, hashJoin=true, joinTime=0,
 actualTotalOutput=61}
            RepeatNode {
                Repeat {
                    PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .
 SimplePathFilter(?1, ?3)) .], {hashJoin=true, estimatedCardinality=50148, indexTime=0,
 joinTime=3}
                }
                Emit {
                    Filter(true)
                }
                LoopsCondition {
```

```
                    LoopsFilter([?1, ?3],eq(2))
                }
            }, annotations={repeatMode=BFS, emitFirst=true, untilFirst=false, leftVar=?
1, rightVar=?3}
        }, finishers=[limit(100)], annotations={path=[Vertex(?1):GraphStep,
 Repeat[Vertex(?3):VertexStep]], joinStats=true, optimizationTime=495, maxVarId=7,
 executionTime=323}
    },
    NeptuneTraverserConverterStep
]

Physical Pipeline
=================
NeptuneGraphQueryStep
    |-- StartOp
    |-- JoinGroupOp
        |-- SpoolerOp(100)
        |-- DynamicJoinOp(PatternNode[(?1, <code>, "AUS", ?) . project ?1 .],
 {estimatedCardinality=1, indexTime=84, hashJoin=true})
        |-- SpoolerOp(100)
        |-- DynamicJoinOp(PatternNode[(?1, <~label>, ?2=<airport>, <~>) . project
 ask .], {estimatedCardinality=3374, indexTime=29, hashJoin=true})
        |-- RepeatOp
            |-- <upstream input> (Iteration 0) [visited=1, output=1 (until=0, emit=1),
 next=1]
            |-- BindingSetQueue (Iteration 1) [visited=61, output=61 (until=0,
 emit=61), next=61]
                |-- SpoolerOp(100)
                |-- DynamicJoinOp(PatternNode[(?3, ?5, ?1, ?6) . project ?
1,?3 . IsEdgeIdFilter(?6) . SimplePathFilter(?1, ?3)) .], {hashJoin=true,
 estimatedCardinality=50148, indexTime=0})
            |-- BindingSetQueue (Iteration 2) [visited=38, output=38 (until=38,
 emit=0), next=0]
                |-- SpoolerOp(100)
                |-- DynamicJoinOp(PatternNode[(?3, ?5, ?1, ?6) . project ?
1,?3 . IsEdgeIdFilter(?6) . SimplePathFilter(?1, ?3)) .], {hashJoin=true,
 estimatedCardinality=50148, indexTime=0})
        |-- LimitOp(100)

Runtime (ms)
============
Query Execution:  392.686
Serialization:   2636.380
```

```
Traversal Metrics
=================
Step                                                               Count  Traversers
    Time (ms)    % Dur
-------------------------------------------------------------------------------------------------
NeptuneGraphQueryStep(Vertex)                                        100         100
      314.162    82.78
NeptuneTraverserConverterStep                                        100         100
       65.333    17.22
                                                >TOTAL                     -           -
      379.495        -

Repeat Metrics
==============
Iteration  Visited   Output    Until     Emit     Next
----------------------------------------------------
        0        1        1        0        1        1
        1       61       61        0       61       61
        2       38       38       38        0        0
----------------------------------------------------
              100      100       38       62       62

Predicates
==========
# of predicates: 16

WARNING: reverse traversal with no edge label(s) - .in() / .both() may impact query
 performance

Results
=======
Count: 100
Output: [v[3], v[3600], v[3614], v[4], v[5], v[6], v[7], v[8], v[9], v[10], v[11],
 v[12], v[47], v[49], v[136], v[13], v[15], v[16], v[17], v[18], v[389], v[20], v[21],
 v[22], v[23], v[24], v[25], v[26], v[27], v[28], v[416], v[29], v[30], v[430], v[31],
 v[9...
Response serializer: GRYO_V3D0
Response size (bytes): 23566

Index Operations
================
Query execution:
    # of statement index ops: 3
    # of unique statement index ops: 3
```

```
    Duplication ratio: 1.0
    # of terms materialized: 0
Serialization:
    # of statement index ops: 200
    # of unique statement index ops: 140
    Duplication ratio: 1.43
    # of terms materialized: 393
```

Selain rencana kueri dikembalikan oleh panggilan ke Neptune `explain`, hasil `profile` menyertakan statistik runtime sekitar eksekusi kueri. Setiap operasi Gabungan ditandai dengan waktu yang dibutuhkan untuk melakukan penggabungannya serta jumlah sebenarnya dari solusi yang melewatinya.

Output `profile` menyertakan waktu yang dibutuhkan selama fase eksekusi kueri inti, serta fase serialisasi jika opsi `profile.serializer` telah ditentukan.

Rincian operasi indeks yang dilakukan selama setiap fase juga disertakan di bagian bawah output `profile`.

Perhatikan bahwa proses berturut-turut dari kueri yang sama dapat menunjukkan hasil yang berbeda dalam hal operasi run-time dan indeks karena caching.

Untuk kueri yang menggunakan langkah `repeat()`, rincian perbatasan pada setiap iterasi tersedia jika langkah `repeat()` didorong ke bawah sebagai bagian dari `NeptuneGraphQueryStep`.

Perbedaan dalam laporan **`profile`** saat DFE diaktifkan

Ketika mesin kueri alternatif Neptune DFE diaktifkan, output `profile` agak berbeda:

Dioptimalkan oleh Traversal: Bagian ini serupa dengan yang ada di output `explain`, tetapi berisi informasi tambahan. Ini menyertakan jenis operator DFE yang dipertimbangkan dalam perencanaan, dan kasus terburuk terkait dan perkiraan biaya kasus terbaik.

Physical Pipeline: Bagian ini menangkap operator yang digunakan untuk mengeksekusi query. `DFESubQuery`elemen abstrak rencana fisik yang digunakan oleh DFE untuk melaksanakan bagian dari rencana yang menjadi tanggung jawabnya. `DFESubQuery`Elemen-elemen tersebut dibuka di bagian berikut di mana statistik DFE terdaftar.

DFEQueryStatistik Mesin: Bagian ini muncul hanya ketika setidaknya sebagian dari kueri dijalankan oleh DFE. Ini menguraikan berbagai statistik runtime yang khusus untuk DFE, dan berisi rincian rinci dari waktu yang dihabiskan di berbagai bagian eksekusi kueri, oleh. `DFESubQuery`

Subquery bersarang dalam `DFESubQuery` elemen yang berbeda diratakan di bagian ini, dan pengidentifikasi unik ditandai dengan header yang dimulai dengan. `subQuery=`

Metrik Traversal: Bagian ini menunjukkan metrik traversal tingkat langkah, dan ketika mesin DFE menjalankan semua atau sebagian kueri, menampilkan metrik untuk dan/atau. `DFEStep` `NeptuneInterleavingStep` Lihat Menyetel kueri Gremlin menggunakan `explain` dan `profile`.

> **Note**
>
> DFE adalah fitur eksperimental yang dirilis dalam mode lab, format tepat output `profile` dapat berubah.

Sampel output **`profile`** ketika Neptune Dataflow engine (DFE) diaktifkan

Ketika mesin DFE sedang digunakan untuk menjalankan query Gremlin, output dari diformat seperti yang API `profile` Gremlin ditunjukkan pada contoh di bawah ini.

Kueri:

```
curl https://your-neptune-endpoint:port/gremlin/profile \
  -d "{\"gremlin\": \"g.withSideEffect('Neptune#useDFE', true).V().has('code',
 'ATL').out()\"}"
```

```
*******************************************************
                    Neptune Gremlin Profile
    *******************************************************

    Query String
    ==================
    g.withSideEffect('Neptune#useDFE', true).V().has('code', 'ATL').out()

    Original Traversal
    ==================
    [GraphStep(vertex,[]), HasStep([code.eq(ATL)]), VertexStep(OUT,vertex)]

    Optimized Traversal
    ===================
    Neptune steps:
    [
        DFEStep(Vertex) {
          DFENode {
```

```
            DFEJoinGroupNode[null](
              children=[
                DFEPatternNode((?1, vp://code[419430926], ?4, defaultGraph[526]) .
project DISTINCT[?1] objectFilters=(in(ATL[452987149]) . ), {rangeCountEstimate=1},
                  opInfo=(type=PipelineJoin,
cost=(exp=(in=1.00,out=1.00,io=0.00,comp=0.00,mem=0.00),wc=(in=1.00,out=1.00,io=0.00,comp=0.00
                    disc=(type=PipelineScan,
cost=(exp=(in=1.00,out=1.00,io=0.00,comp=0.00,mem=34.00),wc=(in=1.00,out=1.00,io=0.00,comp=0.0
                DFEPatternNode((?1, ?5, ?6, ?7) . project ALL[?1, ?6] graphFilters=(!=
defaultGraph[526] . ), {rangeCountEstimate=92233720368547758O7})],
              opInfo=[
                OperatorInfoWithAlternative[
                  rec=(type=PipelineJoin,
cost=(exp=(in=1.00,out=27.76,io=0.00,comp=0.00,mem=0.00),wc=(in=1.00,out=27.76,io=0.00,comp=0.
                    disc=(type=PipelineScan,
cost=(exp=(in=1.00,out=27.76,io=Infinity,comp=0.00,mem=295147905179352830000.00),wc=(in=1.00,o
                  alt=(type=PipelineScan,
cost=(exp=(in=1.00,out=27.76,io=Infinity,comp=0.00,mem=295147905179352830000.00),wc=(in=1.00,o
           } [Vertex(?1):GraphStep, Vertex(?6):VertexStep]
         } ,
         NeptuneTraverserConverterDFEStep,
         DFECleanupStep
     ]


     Physical Pipeline
     =================
     DFEStep
         |-- DFESubQuery1

     DFEQueryEngine Statistics
     =================
     DFESubQuery1

##################################################################################################
     # ID # Out #1 # Out #2 # Name                   # Arguments
                                                                                  # Mode #
Units In # Units Out # Ratio  # Time (ms) #

##################################################################################################
     # 0  # 1      # -      # DFESolutionInjection  # solutions=[]
                                                                                # -    # 0
       # 1         # 0.00   # 0.01      #
```

```
    #    #        #        #                        # outSchema=[]
                                                                  #      #
      #           #        #           #

 ##########################################################################################
    # 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_1 # -    #
 1        # 1         # 1.00   # 0.02      #

 ##########################################################################################
    # 2  # 3      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_2 # -    #
 1        # 242       # 242.00 # 0.02      #

 ##########################################################################################
    # 3  # 4      # -      # DFEMergeChunks        # -
                                                                  # -    # 242
      # 242       # 1.00   # 0.01      #

 ##########################################################################################
    # 4  # -      # -      # DFEDrain              # -
                                                                  # -    # 242
      # 0         # 0.00   # 0.01      #

 ##########################################################################################


    subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/
graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_1

 ##########################################################################################
    # ID # Out #1 # Out #2 # Name                  # Arguments
                          # Mode # Units In # Units Out # Ratio # Time (ms) #

 ##########################################################################################
    # 0  # 1      # -      # DFEPipelineScan      # pattern=Node(?1) with property
 'code' as ?4 and label 'ALL' # -    # 0        # 1         # 0.00  # 0.22      #
    #    #        #        #                      # inlineFilters=[(?4 IN ["ATL"])]
                          #      #          #           #       #           #
    #    #        #        #                      # patternEstimate=1
                          #      #          #           #       #           #

 ##########################################################################################
```

```
   # 1  # 2      # -      # DFEMergeChunks       # -
                        # -    # 1        # 1         # 1.00  # 0.02      #

 ##########################################################################################
   # 2  # 4      # -      # DFERelationalJoin    # joinVars=[]
                        # -    # 2        # 1         # 0.50  # 0.09      #

 ##########################################################################################
   # 3  # 2      # -      # DFESolutionInjection # solutions=[]
                        # -    # 0        # 1         # 0.00  # 0.01      #
   #    #        #        #                      # outSchema=[]
                        #      #          #           #       #           #

 ##########################################################################################
   # 4  # -      # -      # DFEDrain             # -
                        # -    # 1        # 0         # 0.00  # 0.01      #

 ##########################################################################################


   subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/
graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_2

 ##########################################################################################
   # ID # Out #1 # Out #2 # Name                 # Arguments
 # Mode # Units In # Units Out # Ratio  # Time (ms) #

 ##########################################################################################
   # 0  # 1      # -      # DFESolutionInjection # solutions=[]
 # -    # 0        # 1         # 0.00   # 0.01      #
   #    #        #        #                      # outSchema=[?1]
 #      #          #           #        #           #

 ##########################################################################################
   # 1  # 2      # 3      # DFETee               # -
 # -    # 1        # 2         # 2.00   # 0.01      #

 ##########################################################################################
   # 2  # 4      # -      # DFEDistinctColumn    # column=?1
 # -    # 1        # 1         # 1.00   # 0.21      #
   #    #        #        #                      # ordered=false
 #      #          #           #        #           #

 ##########################################################################################
```

```
   # 3  # 5      # -      # DFEHashIndexBuild    # vars=[?1]
# -    # 1        # 1         # 1.00   # 0.03      #

##################################################################################################
   # 4  # 5      # -      # DFEPipelineJoin      # pattern=Edge((?1)-[?7:?5]->(?6))
# -    # 1        # 242       # 242.00 # 0.51      #
   #    #        #         #                     # constraints=[]
#      #          #           #        #           #
   #    #        #         #                     # patternEstimate=9223372036854775807
#      #          #           #        #           #

##################################################################################################
   # 5  # 6      # 7      # DFESync              # -
# -    # 243      # 243       # 1.00   # 0.02      #

##################################################################################################
   # 6  # 8      # -      # DFEForwardValue      # -
# -    # 1        # 1         # 1.00   # 0.01      #

##################################################################################################
   # 7  # 8      # -      # DFEForwardValue      # -
# -    # 242      # 242       # 1.00   # 0.02      #

##################################################################################################
   # 8  # 9      # -      # DFEHashIndexJoin     # -
# -    # 243      # 242       # 1.00   # 0.31      #

##################################################################################################
   # 9  # -      # -      # DFEDrain             # -
# -    # 242      # 0         # 0.00   # 0.01      #

##################################################################################################


   Runtime (ms)
   ============
   Query Execution: 11.744

   Traversal Metrics
   =================
   Step                                                               Count
Traversers       Time (ms)    % Dur

-------------------------------------------------------------------------------------------------
```

```
   DFEStep(Vertex)                                                    242
242          10.849    95.48
   NeptuneTraverserConverterDFEStep                                   242
242           0.514     4.52
                                                  >TOTAL                     -
-          11.363        -

   Predicates
   ==========
   # of predicates: 18

   Results
   =======
   Count: 242


   Index Operations
   ================
   Query execution:
       # of statement index ops: 0
       # of terms materialized: 0
```

> **Note**
>
> Karena mesin DFE adalah fitur eksperimental yang dirilis dalam mode lab, format `profile` output yang tepat dapat berubah.

Menyetel kueri Gremlin menggunakan **`explain`** dan **`profile`**

Anda sering dapat menyetel kueri Gremlin Anda di Amazon Neptunus untuk mendapatkan kinerja yang lebih baik, menggunakan informasi yang tersedia untuk Anda dalam laporan yang Anda dapatkan dari penjelasan dan profil Neptunus. APIs Untuk melakukannya, ia membantu untuk memahami bagaimana Neptune memproses traversals Gremlin.

> **Important**
>
> Perubahan dibuat di TinkerPop versi 3.4.11 yang meningkatkan kebenaran bagaimana kueri diproses, tetapi untuk saat ini terkadang dapat berdampak serius pada kinerja kueri. Misalnya, kueri semacam ini dapat berjalan jauh lebih lambat:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  out()
```

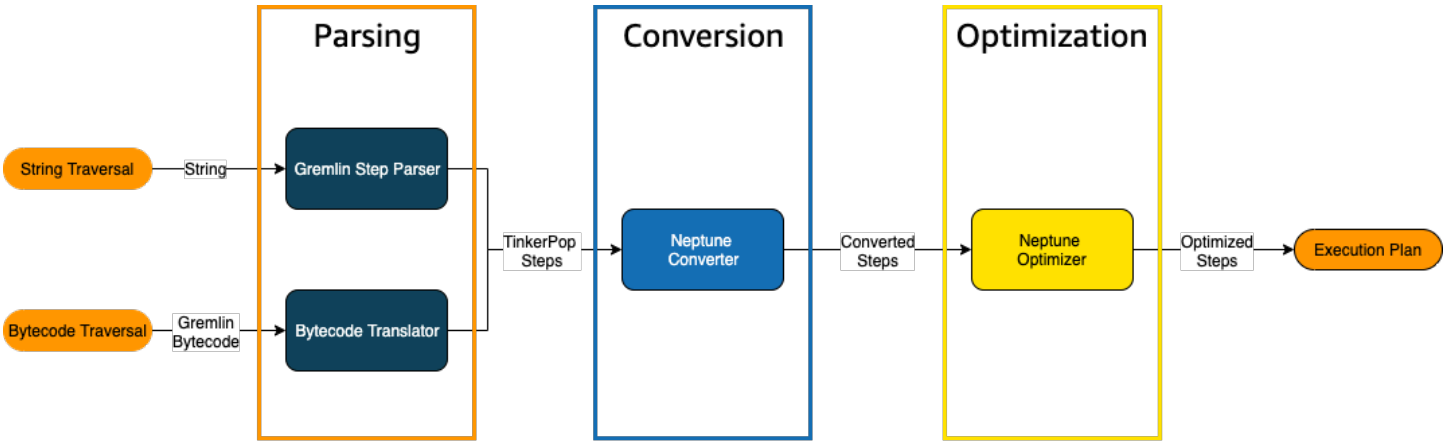
Simpul setelah langkah batas sekarang diambil dengan cara yang tidak optimal karena perubahan 3.4.11. TinkerPop Untuk menghindari hal ini, Anda dapat memodifikasi kueri dengan menambahkan langkah penghalang () kapan saja setelah`order().by()`. Contoh:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

TinkerPop 3.4.11 diaktifkan di mesin Neptunus versi 1.0.5.0.

Memahami pemrosesan traversal Gremlin di Neptune

Ketika traversal Gremlin dikirim ke Neptune, ada tiga proses utama yang mengubah traversal menjadi rencana eksekusi yang mendasari untuk dieksekusi mesin. Ketiganya adalah parsing, conversion, dan optimization:



Proses penguraian traversal

Langkah pertama dalam memproses traversal adalah mengurainya ke dalam bahasa yang sama. Di Neptunus, bahasa umum itu adalah serangkaian langkah yang merupakan bagian TinkerPop dari API. TinkerPop Masing-masing langkah ini merupakan unit komputasi dalam traversal.

Anda dapat mengirim traversal Gremlin ke Neptune baik sebagai string atau sebagai bytecode. Titik akhir REST dan metode `submit()` driver klien Java mengirim traversals sebagai string, seperti dalam contoh ini:

```
client.submit("g.V()")
```

Driver aplikasi dan bahasa yang menggunakan Varian bahasa Gremlin (GLV) mengirim traversals dalam bytecode.

Proses konversi traversal

Langkah kedua dalam memproses traversal adalah mengubah TinkerPop langkah-langkahnya menjadi serangkaian langkah Neptunus yang dikonversi dan tidak dikonversi. Sebagian besar langkah dalam bahasa kueri Apache TinkerPop Gremlin dikonversi ke langkah-langkah khusus Neptunus yang dioptimalkan untuk dijalankan pada mesin Neptunus yang mendasarinya. Ketika sebuah TinkerPop langkah tanpa ekuivalen Neptunus ditemui dalam traversal, langkah itu dan semua langkah selanjutnya dalam traversal diproses oleh mesin kueri. TinkerPop

Untuk informasi selengkapnya tentang langkah-langkah apa yang dapat dikonversi dalam keadaan apa, lihat Dukungan langkah Gremlin.

Proses optimasi traversal

Langkah terakhir dalam pemrosesan traversal adalah menjalankan serangkaian langkah dikonversi dan non-dikonversi melalui optimizer, untuk mencoba menentukan rencana eksekusi terbaik. Output dari optimasi ini adalah rencana eksekusi yang diproses mesin Neptune.

Menggunakan API **`explain`** Gremlin Neptune untuk menyetel kueri

API explain Neptune tidak sama dengan langkah `explain()` Gremlin. Ia mengembalikan rencana eksekusi akhir yang akan diproses mesin Neptune ketika mengeksekusi kueri. Karena tidak melakukan pemrosesan apa pun, ia mengembalikan rencana yang sama terlepas dari parameter yang digunakan, dan outputnya tidak mengandung statistik tentang eksekusi aktual.

Pertimbangkan traversal sederhana berikut yang menemukan semua vertex bandara untuk Anchorage:

```
g.V().has('code','ANC')
```

Ada dua cara untuk menjalankan traversal ini melalui API `explain` Neptune. Cara pertama adalah untuk membuat panggilan REST ke titik akhir explain, seperti ini:

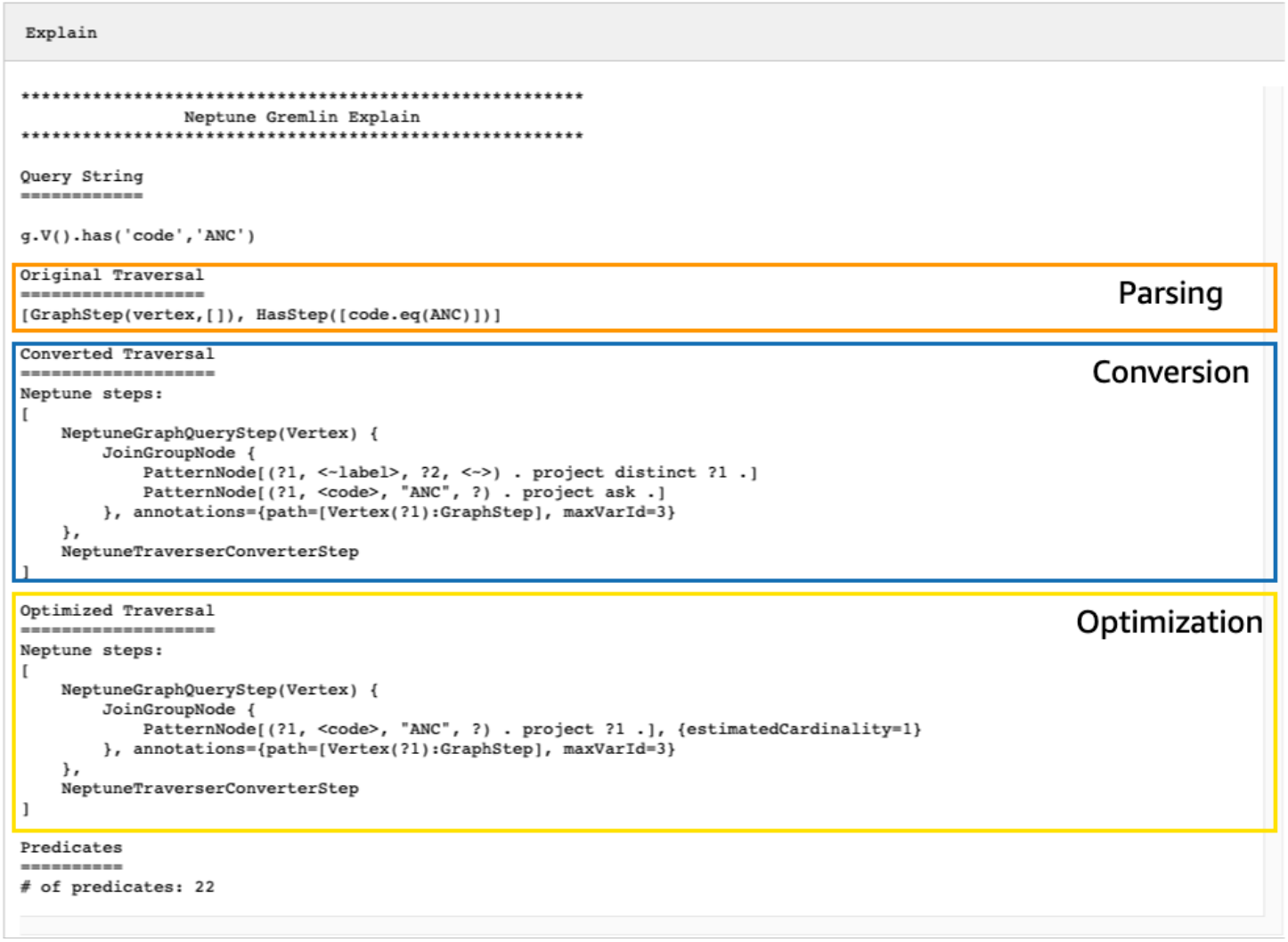
```
curl -X POST https://your-neptune-endpoint:port/gremlin/explain -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

Cara kedua adalah menggunakan workbench cell magic %%gremlin Neptune dengan parameter `explain`. Ini melewati traversal yang terkandung dalam tubuh sel ke API `explain` Neptune dan kemudian menampilkan output yang dihasilkan ketika Anda menjalankan sel:

```
%%gremlin explain

g.V().has('code','ANC')
```

Output `explain` API yang dihasilkan menjelaskan rencana eksekusi Neptunus untuk traversal. Seperti yang bisa Anda lihat pada gambar di bawah ini, rencananya mencakup masing-masing dari 3 langkah dalam alur pemrosesan:



Penyetelan traversal dengan melihat langkah-langkah yang tidak dikonversi

Salah satu hal pertama yang harus dicari di output API `explain` Neptune adalah untuk langkah-langkah Gremlin yang tidak dikonversi ke langkah asli Neptune. Dalam rencana kueri, ketika langkah yang ditemui tidak dapat dikonversi ke langkah asli Neptune, langkah tersebut dan semua langkah berikutnya dalam rencana diproses oleh server Gremlin.

Dalam contoh di atas, semua langkah dalam traversal dikonversi. Mari kita periksa output API `explain` untuk traversal ini:

```
g.V().has('code','ANC').out().choose(hasLabel('airport'), values('code'), constant('Not
 an airport'))
```

Seperti yang bisa Anda lihat pada gambar di bawah ini, Neptune tidak bisa mengonversi langkah `choose()`:

```
Explain
```

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============

g.V().has('code','ANC').out().choose(hasLabel('airport'), values('code'), constant('Not an airport'))

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(OUT,vertex), ChooseStep([HasStep([~label.eq(airport)]), HasNex

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [ChooseStep([HasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [PropertiesSte

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .], {estimatedCardinality=1}
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .], {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [ChooseStep([NeptuneHasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [Proper

WARNING: >> ChooseStep([NeptuneHasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [PropertiesStep([code],value), E

Predicates
==========
# of predicates: 26
```

Ada beberapa hal yang dapat Anda lakukan untuk menyetel performa traversal. Yang pertama adalah menulis ulang sedemikian rupa untuk menghilangkan langkah yang tidak dapat dikonversi. Lainnya adalah memindahkan langkah ke akhir traversal sehingga semua langkah lain dapat dikonversi ke yang asli.

Rencana kueri dengan langkah-langkah yang tidak dikonversi tidak selalu perlu disetel. Jika langkah-langkah yang tidak dapat dikonversi berada di akhir traversal, dan terkait dengan bagaimana output diformat alih-alih bagaimana grafik ditraversalkan, langkah mungkin memiliki sedikit efek pada performa.

Hal lain yang harus dicari ketika memeriksa output dari API `explain` Neptune adalah langkah-langkah yang tidak menggunakan indeks. Traversal berikut menemukan semua bandara dengan penerbangan yang mendarat di Anchorage:

```
g.V().has('code','ANC').in().values('code')
```

Output dari API explain untuk traversal ini adalah:

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============

g.V().has('code','ANC').in().values('code')

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(IN,vertex),
 PropertiesStep([code],value)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
            PatternNode[(?3, ?7, ?8, <~>) . project ?3,?8 . ContainsFilter(?7 in
 (<code>)) .]
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
```

```
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .],
 {estimatedCardinality=1}
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .],
 {estimatedCardinality=INFINITY}
            PatternNode[(?3, ?7=<code>, ?8, <~>) . project ?3,?8 .],
 {estimatedCardinality=7564}
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 26

WARNING: reverse traversal with no edge label(s) - .in() / .both() may impact query
 performance
```

Pesan `WARNING` di bagian bawah output terjadi karena langkah `in()` dalam traversal tidak dapat ditangani menggunakan salah satu dari 3 indeks yang Neptune pertahankan (lihat Bagaimana Pernyataan Diindeks di Neptune dan Pernyataan Gremlin di Neptune). Karena langkah `in()` tidak berisi filter edge, langkah itu tidak dapat diselesaikan menggunakan indeks `SPOG`, `POGS` atau `GPSO`. Sebaliknya, Neptune harus melakukan pemindaian serikat untuk menemukan vertex yang diminta, yang jauh lebih efisien.

Ada dua cara untuk menyetel traversal dalam situasi ini. Yang pertama adalah menambahkan satu atau lebih kriteria penyaringan ke langkah `in()` sehingga pencarian yang diindeks dapat digunakan untuk menyelesaikan kueri. Untuk contoh di atas, ini mungkin:

```
g.V().has('code','ANC').in('route').values('code')
```

Output dari API `explain` Neptune untuk traversal yang direvisi tidak lagi berisi pesan `WARNING`:

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************
```

```
Query String
============

g.V().has('code','ANC').in('route').values('code')

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(IN,[route],vertex),
 PropertiesStep([code],value)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .
 ContainsFilter(?5 in (<route>)) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
            PatternNode[(?3, ?7, ?8, <~>) . project ?3,?8 . ContainsFilter(?7 in
 (<code>)) .]
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .],
 {estimatedCardinality=1}
            PatternNode[(?3, ?5=<route>, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?
6) .], {estimatedCardinality=32042}
            PatternNode[(?3, ?7=<code>, ?8, <~>) . project ?3,?8 .],
 {estimatedCardinality=7564}
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
```

```
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 26
```

Pilihan lain jika Anda menjalankan banyak traversals semacam ini adalah untuk menjalankan traversal tersebut dalam sebuah klaster DB Neptune yang memiliki indeks OSGP opsional diaktifkan (lihat Mengaktifkan Indeks OSGP). Mengaktifkan indeks OSGP memiliki kelemahan:

- Indeks ini harus diaktifkan dalam klaster DB sebelum data dimuat.
- Tingkat penyisipan untuk vertex dan edge dapat melambat hingga 23%.
- Penggunaan penyimpanan akan meningkat sekitar 20%.
- Kueri Baca yang menyebarkan permintaan di semua indeks mungkin telah meningkatkan latensi.

Memiliki indeks OSGP lebih masuk akal untuk serangkaian pola kueri terbatas, tetapi kecuali jika Anda sering menjalankannya, biasanya lebih baik untuk mencoba untuk memastikan bahwa traversals yang Anda tulis dapat diselesaikan menggunakan tiga indeks utama.

Menggunakan jumlah besar predikat

Neptune memperlakukan setiap label edge dan setiap nama properti vertex atau edge yang berbeda dalam grafik Anda sebagai predikat, dan dirancang secara default untuk bekerja dengan jumlah predikat berbeda yang relatif rendah. Bila Anda memiliki lebih dari beberapa ribu predikat dalam data grafik Anda, performa dapat menurun.

Output `explain` Neptune akan memperingatkan Anda jika hal ini terjadi:

```
Predicates
==========
# of predicates: 9549
WARNING: high predicate count (# of distinct property names and edge labels)
```

Jika tidak nyaman untuk mengulang pengerjaan model data Anda untuk mengurangi jumlah label dan properti, dan karenanya jumlah predikatnya, cara terbaik untuk menyetel traversal adalah menjalankannya dalam sebuah klaster DB yang memiliki indeks OSGP diaktifkan, seperti yang dibahas di atas.

Menggunakan API **`profile`** Gremlin Neptune untuk menyetel traversal

API `profile` Neptune sangat berbeda dari langkah `profile()` Gremlin. Seperti halnya API `explain`, outputnya mencakup rencana kueri yang digunakan mesin Neptune saat mengeksekusi traversal. Selain itu, output `profile` menyertakan statistik eksekusi aktual untuk traversal, mengingat bagaimana parameternya ditetapkan.

Sekali lagi, ambil traversal sederhana yang menemukan semua vertex bandara untuk Anchorage:

```
g.V().has('code','ANC')
```

Seperti halnya dengan API `explain`, Anda dapat memanggil API `profile` menggunakan panggilan REST:

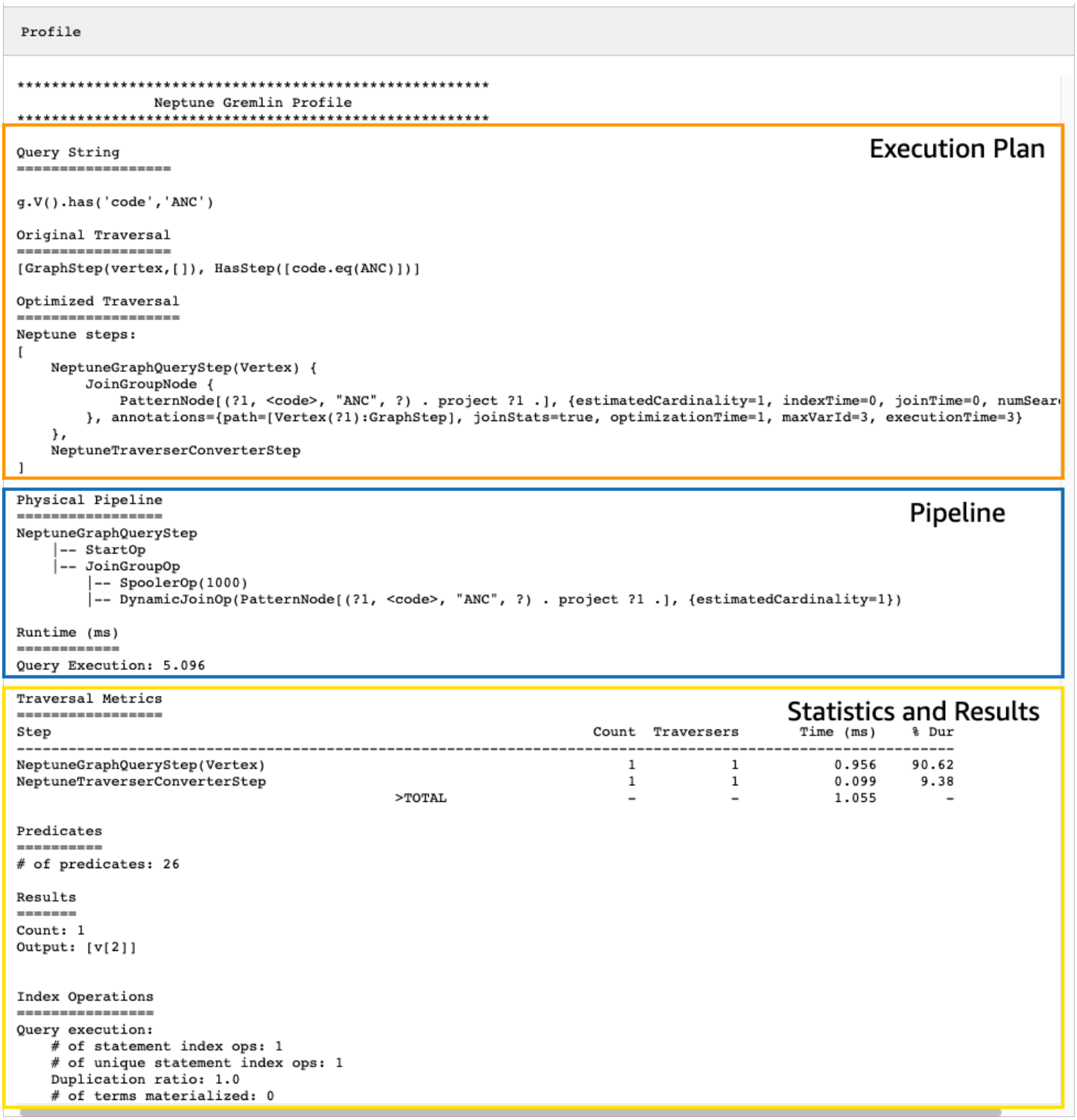
```
curl -X POST https://your-neptune-endpoint:port/gremlin/profile -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

Anda juga menggunakan workbench cell magic %%gremlin Neptune dengan parameter `profile`. Ini melewati traversal yang terkandung dalam tubuh sel ke API `profile` Neptune dan kemudian menampilkan output yang dihasilkan ketika Anda menjalankan sel:

```
%%gremlin profile

g.V().has('code','ANC')
```

Output `profile` API yang dihasilkan berisi rencana eksekusi Neptunus untuk traversal dan statistik tentang eksekusi rencana, seperti yang Anda lihat pada gambar ini:



Dalam output `profile`, bagian rencana eksekusi hanya berisi rencana eksekusi akhir untuk traversal, bukan langkah menengahnya. Bagian alur berisi operasi alur fisik yang dilakukan serta waktu aktual (dalam milidetik) yang diperlukan eksekusi traversal. Metrik runtime sangat membantu

dalam membandingkan waktu yang diperlukan dua versi traversal yang berbeda saat Anda mengoptimalkan mereka.

> **Note**
>
> Runtime awal traversal umumnya lebih panjang dari runtime berikutnya, karena yang pertama menyebabkan data yang relevan untuk di-cache.

Bagian ketiga dari output `profile` berisi statistik eksekusi dan hasil traversal. Untuk melihat bagaimana informasi ini dapat berguna dalam menyetel traversal, pertimbangkan traversal berikut, yang menemukan setiap bandara yang namanya dimulai dengan “Anchora”, dan semua bandara yang dapat dijangkau dalam dua hop dari bandara tersebut, mengembalikan kode bandara, rute penerbangan, dan jarak:

```
%%gremlin profile

g.withSideEffect("Neptune#fts.endpoint", "{your-OpenSearch-endpoint-URL").
    V().has("city", "Neptune#fts Anchora~").
    repeat(outE('route').inV().simplePath()).times(2).
    project('Destination', 'Route').
        by('code').
        by(path().by('code').by('dist'))
```

Metrik traversal di output API **`profile`** Neptune

Set pertama metrik yang tersedia di semua output `profile` adalah metrik traversal. Ini mirip dengan metrik langkah `profile()` Gremlin, dengan beberapa perbedaan:

```
Traversal Metrics
=================
Step                                                               Count  Traversers
    Time (ms)    % Dur
-------------------------------------------------------------------------------------------------
NeptuneGraphQueryStep(Vertex)                                       3856        3856
       91.701     9.09
NeptuneTraverserConverterStep                                       3856        3856
       38.787     3.84
ProjectStep([Destination, Route],[value(code), ...                  3856        3856
      878.786    87.07
```

```
  PathStep([value(code), value(dist)])                                3856        3856
     601.359
                                                >TOTAL                    -           -
    1009.274        -
```

Kolom pertama dari tabel traversal-metrik mencantumkan langkah-langkah yang dijalankan oleh traversal. Dua langkah pertama umumnya adalah langkah-langkah khusus Neptune, `NeptuneGraphQueryStep` dan `NeptuneTraverserConverterStep`.

`NeptuneGraphQueryStep` mewakili waktu eksekusi untuk seluruh bagian dari traversal yang dapat dikonversi dan dieksekusi secara native oleh mesin Neptune.

`NeptuneTraverserConverterStep`mewakili proses mengubah output dari langkah-langkah yang dikonversi tersebut menjadi TinkerPop traversers yang memungkinkan langkah-langkah yang tidak dapat dikonversi langkah-langkah, jika ada, untuk diproses, atau untuk mengembalikan hasil dalam format -kompatibel. TinkerPop

Pada contoh di atas, kita memiliki beberapa langkah yang tidak dikonversi, jadi kita melihat bahwa masing-masing TinkerPop langkah ini (`ProjectStep`,`PathStep`) kemudian muncul sebagai baris dalam tabel.

Kolom kedua dalam tabel`Count`,, melaporkan jumlah pelintas yang diwakili yang melewati langkah, sedangkan kolom ketiga,`Traversers`, melaporkan jumlah pelintas yang melewati langkah itu, seperti yang dijelaskan dalam dokumentasi langkah profil. TinkerPop

Dalam contoh kita ada 3.856 simpul dan 3.856 traversers dikembalikan oleh`NeptuneGraphQueryStep`, dan angka-angka ini tetap sama sepanjang pengolahan yang tersisa karena`ProjectStep`dan`PathStep`memformat hasil, tidak menyaring mereka.

> **Note**
>
> Tidak seperti TinkerPop, mesin Neptunus tidak mengoptimalkan kinerja dengan menumpuk di langkah-langkahnya. `NeptuneGraphQueryStep` `NeptuneTraverserConverterStep` Bulking adalah TinkerPop operasi yang menggabungkan pelintas pada simpul yang sama untuk mengurangi overhead operasional, dan itulah yang menyebabkan dan angka berbeda. `Count` `Traversers` Karena bulking hanya terjadi pada langkah-langkah yang didelegasikan Neptunus, dan bukan pada langkah-langkah yang TinkerPop ditangani Neptunus secara asli, kolom dan kolom jarang berbeda. `Count` `Traverser`

Kolom Waktu melaporkan jumlah milidetik yang diambil langkah, dan kolom `% Dur` melaporkan berapa persen dari total waktu pemrosesan yang diambil langkah. Ini adalah metrik yang memberi tahu Anda tempat untuk memfokuskan upaya penyetelan dengan menunjukkan langkah-langkah yang memakan waktu paling lama.

Metrik operasi indeks di output API **`profile`** Neptune

Satu set metrik dalam output dari API profil Neptune adalah operasi indeks:

```
Index Operations
================
Query execution:
    # of statement index ops: 23191
    # of unique statement index ops: 5960
    Duplication ratio: 3.89
    # of terms materialized: 0
```

Laporan ini:

- Jumlah total pencarian indeks.
- Jumlah pencarian indeks unik yang dilakukan.
- Rasio total pencarian indeks ke yang unik. Rasio yang lebih rendah menunjukkan redundansi yang lebih sedikit.
- Jumlah istilah terwujud dari kamus istilah.

Metrik repeat di output API **`profile`** Neptune

Jika traversal Anda menggunakan `repeat()` seperti pada contoh di atas, maka bagian yang berisi metrik repeat muncul di output `profile`:

```
Repeat Metrics
==============
Iteration  Visited   Output    Until     Emit     Next
--------------------------------------------------------
        0        2        0        0        0        2
        1       53        0        0        0       53
        2     3856     3856     3856        0        0
--------------------------------------------------------
              3911     3856     3856        0       55
```

Laporan ini:

- Jumlah loop untuk satu baris (kolom `Iteration`).
- Jumlah elemen yang dikunjungi oleh loop (kolom `Visited`).
- Jumlah elemen yang dioutput oleh loop (kolom `Output`).
- Elemen terakhir yang dioutput oleh loop (kolom `Until`).
- Jumlah elemen yang dikeluarkan oleh loop (kolom `Emit`).
- Jumlah elemen yang dilewatkan dari loop ke loop berikutnya (kolom `Next`).

Metrik repeat ini sangat membantu dalam memahami faktor percabangan traversal Anda, untuk mendapatkan kesan berapa banyak pekerjaan yang sedang dilakukan oleh database. Anda dapat menggunakan angka-angka ini untuk mendiagnosis masalah performa, terutama ketika traversal yang sama berkinerja berbeda secara dramatis dengan parameter lainnya.

Metrik pencarian teks lengkap di output API **`profile`** Neptune

Ketika sebuah traversal menggunakan pencarian teks penuh seperti dalam contoh di atas, maka bagian yang berisi metrik pencarian teks lengkap (FTS) muncul di output `profile`:

```
FTS Metrics
==============
SearchNode[(idVar=?1, query=Anchora~, field=city) . project ?1 .],
    {endpoint=your-OpenSearch-endpoint-URL, incomingSolutionsThreshold=1000,
 estimatedCardinality=INFINITY,
    remoteCallTimeSummary=[total=65, avg=32.500000, max=37, min=28],
    remoteCallTime=65, remoteCalls=2, joinTime=0, indexTime=0, remoteResults=2}

    2 result(s) produced from SearchNode above
```

Ini menunjukkan kueri yang dikirim ke klaster ElasticSearch (ES) dan melaporkan beberapa metrik tentang interaksi yang dapat membantu Anda menentukan masalah kinerja ElasticSearch yang berkaitan dengan penelusuran teks lengkap:

- Ringkasan informasi tentang panggilan ke ElasticSearch indeks:
  - Jumlah total milidetik yang dibutuhkan oleh semua remoteCalls untuk memenuhi kueri (`total`).
  - Jumlah rata-rata milidetik yang dihabiskan di remoteCall (`avg`).
  - Jumlah minimum milidetik yang dihabiskan di remoteCall (`min`).

  - Jumlah maksimum milidetik yang dihabiskan di remoteCall (`max`).
- Total waktu yang dikonsumsi oleh RemoteCalls to ElasticSearch ()`remoteCallTime`.
- Jumlah RemoteCalls dibuat untuk ElasticSearch ()`remoteCalls`.
- Jumlah milidetik yang dihabiskan dalam gabungan ElasticSearch hasil ()`joinTime`.
- Jumlah milidetik yang dihabiskan dalam pencarian indeks (`indexTime`).
- Jumlah total hasil yang dikembalikan oleh ElasticSearch (`remoteResults`).

Dukungan langkah Gremlin asli di Amazon Neptune

Mesin Amazon Neptune saat ini tidak memiliki dukungan asli penuh untuk semua langkah Gremlin, seperti yang dijelaskan di Menyetel kueri Gremlin. Dukungan saat ini terbagi dalam empat kategori:

- Langkah-langkah Gremlin yang selalu dapat dikonversi ke operasi mesin Neptune asli
- Langkah Gremlin yang dapat dikonversi ke operasi mesin Neptune asli dalam beberapa kasus
- Langkah Gremlin yang tidak pernah dikonversi ke operasi mesin Neptune asli
- Langkah-langkah Gremlin yang tidak didukung di Neptune sama sekali

Langkah-langkah Gremlin yang selalu dapat dikonversi ke operasi mesin Neptune asli

Banyak langkah Gremlin dapat dikonversi ke operasi mesin Neptune asli selama mereka memenuhi kondisi berikut:

- Mereka tidak didahului di dalam kueri dengan langkah yang tidak dapat dikonversi.
- Langkah induk mereka, jika ada, dapat dikonversi,
- Semua traversals turunan mereka, jika ada, dapat dikonversi.

Langkah-langkah Gremlin berikut selalu dikonversi ke operasi mesin Neptune asli jika mereka memenuhi kondisi tersebut:

- dan
- sebagai ()
- menghitung ()
- E ()
- memancarkan ()

- menjelaskan ()
- kelompok ()
- GroupCount ()
- identitas ()
- adalah
- kunci ()
- label ()
- batas ()
- lokal ()
- loop ()
- tidak ()
- atau ()
- profil ()
- properti ()
- subgraf ()
- sampai ()
- V ()
- nilai ()
- ValueMap ()
- nilai ()

Langkah Gremlin yang dapat dikonversi ke operasi mesin Neptune asli dalam beberapa kasus

Beberapa langkah Gremlin dapat dikonversi ke operasi mesin Neptune asli dalam beberapa situasi tetapi tidak pada situasi lain:

- addE( ) — Langkah `addE()` umumnya dapat dikonversi ke operasi mesin Neptune asli, kecuali segera diikuti oleh langkah `property()` yang berisi traversal sebagai kunci.
- addV( ) — Langkah `addV()` umumnya dapat dikonversi ke operasi mesin Neptune asli, kecuali segera diikuti oleh langkah `property()` yang berisi traversal sebagai kunci, atau kecuali beberapa label ditetapkan.

- aggregate( ) — Langkah `aggregate()` umumnya dapat dikonversi ke operasi mesin Neptune asli, kecuali langkah ini digunakan dalam traversal turunan atau sub-traversal, atau kecuali nilai yang disimpan adalah sesuatu selain vertex, edge, id, label atau nilai properti.

  Dalam contoh di bawah ini, `aggregate()` tidak dikonversi karena sedang digunakan dalam traversal turunan:

```
g.V().has('code','ANC').as('a')
     .project('flights').by(select('a')
     .outE().aggregate('x'))
```

  Dalam contoh ini, aggregate( ) tidak dikonversi karena yang disimpan adalah `min()` dari nilai:

```
g.V().has('code','ANC').outE().aggregate('x').by(values('dist').min())
```

- barrier( ) — Langkah `barrier()` umumnya dapat dikonversi ke operasi mesin Neptune asli, kecuali langkah yang mengikutinya tidak dikonversi.
- cap( ) — Satu-satunya kasus di mana langkah `cap()` dikonversi adalah ketika dikombinasikan dengan langkah `unfold()` untuk mengembalikan versi tidak dilipat dari agregat vertex, edge, id, atau nilai poperti. Dalam contoh ini, `cap()` akan dikonversi karena diikuti oleh `.unfold()`:

```
g.V().has('airport','country','IE').aggregate('airport').limit(2)
     .cap('airport').unfold()
```

  Namun, jika Anda menghapus `.unfold()`, `cap()` tidak akan dikonversi:

```
g.V().has('airport','country','IE').aggregate('airport').limit(2)
     .cap('airport')
```

- coalesce () — Satu-satunya kasus di mana `coalesce()` langkah dikonversi adalah ketika mengikuti pola Upsert yang direkomendasikan pada halaman resep. TinkerPop Pola coalesce( ) lainnya tidak diperbolehkan. Konversi terbatas pada kasus di mana semua traversals turunan dapat dikonversi, mereka semua menghasilkan jenis yang sama sebagai output (vertex, edge, id, nilai, kunci, atau label), mereka semua melakukan traversal ke elemen baru, dan mereka tidak mengandung langkah `repeat()`.
- constant( ) — Langkah constant( ) saat ini hanya dikonversi jika digunakan dalam bagian `sack().by()` dari traversal untuk menetapkan nilai konstan, seperti ini:

```
g.V().has('code','ANC').sack(assign).by(constant(10)).out().limit(2)
```

- cyclicPath( ) — Langkah `cyclicPath()` umumnya dapat dikonversi ke operasi mesin Neptune asli, kecuali langkah ini digunakan dengan modulator `by()`, `from()`, atau `to()`. Dalam kueri berikut, misalnya, `cyclicPath()` tidak dikonversi:

```
g.V().has('code','ANC').as('a').out().out().cyclicPath().by('code')
g.V().has('code','ANC').as('a').out().out().cyclicPath().from('a')
g.V().has('code','ANC').as('a').out().out().cyclicPath().to('a')
```

- drop( ) — Langkah `drop()` umumnya dapat dikonversi ke operasi mesin Neptune asli, kecuali langkah ini digunakan di dalam langkah `sideEffect(` atau `optional()`.
- fold () — Hanya ada dua situasi di mana langkah fold () dapat dikonversi, yaitu ketika digunakan dalam pola Upsert yang direkomendasikan pada halaman TinkerPop resep, dan ketika digunakan dalam `group().by()` konteks seperti ini:

```
g.V().has('code','ANC').out().group().by().by(values('code', 'city').fold())
```

- has () — Langkah `has () `umumnya dapat dikonversi ke operasi mesin Neptunus asli yang disediakan kueri dengan `T` menggunakan predikat `P.eq`, `P.neq` atau `P.contains`. Harapkan variasi `has () `yang menyiratkan contoh `P` untuk dikonversi ke native juga, seperti `HasiD ('id1234')` yang setara dengan `has (eq, t.id, 'id1234') `.
- id( ) — Langkah `id()` dikonversi kecuali digunakan pada properti, seperti ini:

```
g.V().has('code','ANC').properties('code').id()
```

- Mergee () — `mergeE()` Langkah dapat dikonversi ke operasi mesin Neptunus asli jika parameter (kondisi gabungan, `onCreate` dan`onMatch`) konstan (baik, konstanta`null`, atau a). `Map` `select()` `Map` Semua contoh di tepi atas dapat dikonversi.
- MergeV () — Langkah mergeV () dapat dikonversi ke operasi mesin Neptunus asli jika parameter (kondisi gabungan, `onCreate` dan`onMatch`) konstan (baik, konstanta, atau `null` a). `Map` `select()` `Map` Semua contoh dalam simpul upserting dapat dikonversi.
- order( ) — Langkah `order()` umumnya dapat dikonversi ke operasi mesin Neptune asli, kecuali langkah yang mengikutinya adalah true:
  - Langkah `order()` berada dalam traversal turunan nested, seperti ini:

```
g.V().has('code','ANC').where(V().out().order().by(id))
```

- Pengurutan lokal sedang digunakan, seperti misalnya dengan `order(local)`.
- Sebuah pembanding kustom sedang digunakan dalam modulasi `by()` untuk mengurutkan. Contohnya adalah penggunaan `sack()` ini:

```
g.withSack(0).
  V().has('code','ANC').
      repeat(outE().sack(sum).by('dist').inV()).times(2).limit(10).
      order().by(sack())
```

- Ada beberapa pengurutan pada elemen yang sama.

- project( ) — Langkah `project()` umumnya dapat dikonversi ke operasi mesin Neptune asli, kecuali jumlah pernyataan `by()` yang mengikuti `project()` tidak cocok dengan jumlah label yang ditentukan, seperti di sini:

```
g.V().has('code','ANC').project('x', 'y').by(id)
```

- range( ) — Langkah `range()` hanya dikonversi ketika ujung bawah rentang yang dimaksud adalah nol (misalnya, `range(0,3)`).
- repeat( ) — Langkah `repeat()` umumnya dapat dikonversi ke operasi mesin Neptune asli, kecuali di-nested di dalam langkah `repeat()` lain, seperti ini:

```
g.V().has('code','ANC').repeat(out().repeat(out()).times(2)).times(2)
```

- sack( ) — Langkah `sack()` umumnya dapat dikonversi ke operasi mesin Neptune asli, kecuali dalam kasus berikut:
  - Jika operator sack non-numerik sedang digunakan.
  - Jika operator sack numerik selain `+`, `-`, `mult`, `div`, `min` dan `max` sedang digunakan.
  - Jika `sack()` digunakan di dalam langkah `where()` untuk menyaring berdasarkan nilai sack, seperti di sini:

```
g.V().has('code','ANC').sack(assign).by(values('code')).where(sack().is('ANC'))
```

- sum( ) — Langkah `sum()` umumnya dapat dikonversi ke operasi mesin Neptune asli, tetapi tidak ketika digunakan untuk menghitung penjumlahan global, seperti ini:

```
g.V().has('code','ANC').outE('routes').values('dist').sum()
```

- union () — `union()` Langkah ini dapat dikonversi ke operasi mesin Neptunus asli selama itu adalah langkah terakhir dalam kueri selain dari langkah terminal.
- unfold () — `unfold()` Langkah ini hanya dapat dikonversi ke operasi mesin Neptunus asli ketika digunakan dalam pola Upsert yang direkomendasikan pada TinkerPop halaman resep, dan ketika digunakan bersama dengan seperti ini: `cap()`

```
g.V().has('airport','country','IE').aggregate('airport').limit(2)
     .cap('airport').unfold()
```

- where( ) — Langkah `where()` umumnya dapat dikonversi ke operasi mesin Neptune asli, kecuali dalam kasus berikut:
  - Saat modulasi by( ) digunakan, seperti ini:

    ```
    g.V().hasLabel('airport').as('a')
         .where(gt('a')).by('runways')
    ```

  - Ketika operator perbandingan selain `eq`, `neq`, `within`, dan `without` digunakan.
  - Ketika agregasi yang disediakan pengguna digunakan.

Langkah Gremlin yang tidak pernah dikonversi ke operasi mesin Neptune asli

Langkah-langkah Gremlin berikut didukung di Neptune tetapi tidak pernah dikonversi ke operasi mesin Neptune asli. Sebaliknya, mereka dieksekusi oleh server Gremlin.

- pilih ()
- koin ()
- menyuntikkan ()
- cocok ()
- matematika ()
- maks ()
- berarti ()
- min ()
- pilihan ()

- opsional ()
- jalan ()
- PropertyMap ()
- sampel ()
- lewati ()
- ekor ()
- TimeLimit ()
- pohon ()

Langkah-langkah Gremlin yang tidak didukung di Neptune sama sekali

Langkah-langkah Gremlin berikut tidak didukung sama sekali di Neptune. Dalam kebanyakan kasus ini karena mereka memerlukan `GraphComputer`, yang saat ini tidak didukung Neptune.

- ConnectedComponent ()
- io ()
- ShortestPath ()
- denganKomputer ()
- PageRank ()
- PeerPressure ()
- program ()

`io()`Langkah ini sebenarnya sebagian didukung, karena dapat digunakan untuk `read()` dari URL tetapi tidak untuk`write()`.

Menggunakan Gremlin dengan mesin kueri Neptunus DFE

Jika Anda mengaktifkan mesin kueri alternatif Neptunus yang dikenal sebagai DFE dalam mode lab (dengan menyetel parameter cluster `neptune_lab_mode` DB ke)`DFEQueryEngine=enabled`, maka Neptunus menerjemahkan Gremlin hanya-baca queries/traversals menjadi representasi logis perantara dan menjalankannya pada mesin DFE bila memungkinkan.

Namun, DFE belum mendukung semua langkah Gremlin. Ketika sebuah langkah tidak dapat dijalankan secara asli di DFE, Neptunus kembali menjalankan langkahnya. TinkerPop `profile`Laporan `explain` dan termasuk peringatan ketika ini terjadi.

Cakupan langkah Gremlin di DFE

Gremlin DFE adalah fitur labmode dan dapat digunakan dengan mengaktifkan parameter cluster atau menggunakan petunjuk kueri. `Neptune#useDFE` Untuk informasi lebih lanjut, silakan merujuk ke Menggunakan Gremlin dengan mesin kueri Neptunus DFE.

Langkah-langkah berikut tersedia untuk digunakan di Gremlin DFE.

Langkah-langkah jalur dan traversal:

asDate () , barrier (), call (), cap (), dateAdd (), DateDiff (), disjunct (), drop (), gagal (), filter (), flatMap (), id (), identity (), index (), intersect (), inject (), label (), length (), loop (), map (), order ()), pesanan (lokal), jalur (), project (), range (), repeat (), reverse (), sack (), sample (), select (), sideEffect (), split (), unfold (), union ()

Langkah-langkah agregat dan pengumpulan:

agregat (global), gabungkan (), hitung (), dedup (), dedup (lokal), lipat (), grup (), groupCount (),

Langkah-langkah matematika:

max (), mean (), min (), sum ()

Langkah-langkah elemen:

otherV (), elementMap (), elemen (), v (), out (), in (), keduanya (), oute (), inE (), boThe (), outV (), inV (), bothV (), otherV ()

Langkah-langkah properti:

properti (), key (), valueMap (), value ()

Langkah-langkah filter:

dan (), coalesce (), coin (), has (), is (), local (), none (), not (), or (), where ()

Langkah-langkah manipulasi string:

concat (), lTrim (), rTrim (), substring (), toLower (),toupper (), trim ()

Predikat:

- Bandingkan: eq, neq, lt, lte, gt, gte

- Berisi: di dalam, tanpa
- TextP: EndingWith, berisi,,, NotContaining notStartingWith notEndingWith
- P: dan, atau, antara, di luar, di dalam

Batasan

Ulangi dengan Batas, Label di dalam traversal berulang dan dedup belum didukung di DFE.

```
// With Limit inside the repeat traversal
  g.V().has('code','AGR').repeat(out().limit(5)).until(has('code','FRA'))

  // With Labels inside the repeat traversal
  g.V().has('code','AGR').repeat(out().as('a')).until(has('code','FRA'))

  // With Dedup inside the repeat traversal
  g.V().has('code','AGR').repeat(out().dedup()).until(has('code','FRA'))
```

Jalur dengan pengulangan bersarang, atau langkah percabangan belum didukung.

```
// Path with branching steps
  g.V().has('code','AGR').union(identity, outE().inV()).path().by('code')


  // With nested repeat
  g.V().has('code','AGR').repeat(out().union(identity(), out())).path().by('code')
```

Perencanaan kueri interleaving

Ketika proses penerjemahan menemukan langkah Gremlin yang tidak memiliki operator DFE asli yang sesuai, sebelum kembali menggunakan Tinkerpop, ia mencoba menemukan bagian kueri perantara lainnya yang dapat dijalankan secara asli di mesin DFE. Ini dilakukan dengan menerapkan logika interleaving ke traversal tingkat atas. Hasilnya adalah langkah-langkah yang didukung digunakan sedapat mungkin.

Terjemahan kueri non-awalan menengah seperti itu direpresentasikan menggunakan `NeptuneInterleavingStep` dalam `explain` dan `profile` output.

Untuk perbandingan kinerja, Anda mungkin ingin mematikan interleaving dalam kueri, sambil tetap menggunakan mesin DFE untuk menjalankan bagian awalan. Atau, Anda mungkin hanya ingin menggunakan TinkerPop mesin untuk eksekusi kueri non-awalan. Anda dapat melakukan ini dengan menggunakan petunjuk `disableInterleaving` kueri.

Sama seperti petunjuk useDFE kueri dengan nilai `false` mencegah kueri dijalankan di DFE sama sekali, petunjuk `disableInterleaving` kueri dengan nilai `true` mematikan interleaving DFE untuk terjemahan kueri. Contoh:

```
g.with('Neptune#disableInterleaving', true)
 .V().has('genre','drama').in('likes')
```

Diperbarui Gremlin **`explain`** dan output **`profile`**

Gremlin menjelaskan memberikan rincian tentang traversal yang dioptimalkan yang digunakan Neptunus untuk menjalankan kueri. Lihat contoh `explain` keluaran DFE untuk contoh seperti apa `explain` output saat mesin DFE diaktifkan.

API `profile` GremlinMenjalankan traversal Gremlin tertentu, mengumpulkan berbagai metrik tentang proses, dan menghasilkan laporan profil yang berisi detail tentang rencana kueri yang dioptimalkan dan statistik runtime dari berbagai operator. Lihat contoh `profile` keluaran DFE untuk contoh seperti apa `profile` output saat mesin DFE diaktifkan.

> **Note**
>
> Karena mesin DFE adalah fitur eksperimental yang dirilis dalam mode lab, format `explain` dan `profile` output yang tepat dapat berubah.

Mengakses Grafik Neptunus dengan OpenCypher

Neptunus mendukung pembuatan aplikasi grafik menggunakan OpenCypher, saat ini salah satu bahasa kueri paling populer untuk pengembang yang bekerja dengan database grafik. Pengembang, analis bisnis, dan ilmuwan data menyukai sintaks yang terinspirasi SQL OpenCypher karena menyediakan struktur yang akrab untuk menyusun kueri untuk aplikasi grafik.

OpenCypher adalah bahasa query deklaratif untuk grafik properti yang awalnya dikembangkan oleh Neo4j, kemudian open-source pada tahun 2015, dan berkontribusi pada proyek OpenCypher di

bawah lisensi open-source Apache 2. Sintaksnya didokumentasikan dalam Referensi Bahasa Kueri Cypher, Versi 9.

Untuk keterbatasan dan perbedaan dukungan Neptunus dari spesifikasi OpenCypher, lihat. Kepatuhan spesifikasi OpenCypher di Amazon Neptunus

> **Note**
>
> Implementasi Neo4j saat ini dari bahasa kueri Cypher telah menyimpang dalam beberapa hal dari spesifikasi OpenCypher. Jika Anda memigrasikan kode Neo4j Cypher saat ini ke Neptunus, lihat dan untuk bantuan. Kompatibilitas Neptunus dengan Neo4j Menulis ulang kueri Cypher untuk dijalankan di OpenCypher di Neptunus

Dimulai dengan rilis mesin 1.1.1.0, OpenCypher tersedia untuk penggunaan produksi di Neptunus.

Gremlin vs OpenCypher: persamaan dan perbedaan

Gremlin dan OpenCypher keduanya adalah bahasa kueri grafik properti, dan keduanya saling melengkapi dalam banyak hal.

Gremlin dirancang untuk menarik programmer dan cocok dengan kode. Akibatnya, Gremlin sangat penting oleh desain, sedangkan sintaks deklaratif OpenCypher mungkin terasa lebih akrab bagi orang-orang dengan pengalaman SQL atau SPARQL. Gremlin mungkin tampak lebih alami bagi ilmuwan data yang menggunakan Python di notebook Jupyter, sedangkan OpenCypher mungkin tampak lebih intuitif bagi pengguna bisnis dengan beberapa latar belakang SQL.

Yang menyenangkan adalah Anda tidak harus memilih antara Gremlin dan OpenCypher di Neptunus. Kueri dalam salah satu bahasa dapat beroperasi pada grafik yang sama terlepas dari mana dari dua bahasa yang digunakan untuk memasukkan data tersebut. Anda mungkin merasa lebih nyaman menggunakan Gremlin untuk beberapa hal dan OpenCypher untuk orang lain, tergantung pada apa yang Anda lakukan.

Gremlin menggunakan sintaks imperatif yang memungkinkan Anda mengontrol bagaimana Anda bergerak melalui grafik Anda dalam serangkaian langkah, yang masing-masing mengambil aliran data, melakukan beberapa tindakan di atasnya (menggunakan filter, peta, dan sebagainya), dan kemudian output hasilnya ke langkah berikutnya. Kueri Gremlin biasanya mengambil formulir`g.V()`, diikuti dengan langkah-langkah tambahan.

Di OpenCypher, Anda menggunakan sintaks deklaratif, terinspirasi oleh SQL, yang menentukan pola node dan hubungan untuk ditemukan dalam grafik Anda menggunakan sintaks motif (seperti). `()-[]->()` Query OpenCypher sering dimulai dengan `MATCH` klausa, diikuti oleh klausa lain seperti,, dan. `WHERE` `WITH` `RETURN`

Memulai menggunakan OpenCypher

Anda dapat menanyakan data grafik properti di Neptunus menggunakan OpenCypher terlepas dari bagaimana itu dimuat, tetapi Anda tidak dapat menggunakan OpenCypher untuk menanyakan data yang dimuat sebagai RDF.

Pemuat massal Neptunus menerima data grafik properti dalam format CSV untuk Gremlin, dan dalam format CSV untuk OpenCypher. Juga, tentu saja, Anda dapat menambahkan data properti ke grafik Anda menggunakan kueri Gremlin and/or OpenCypher.

Ada banyak tutorial online yang tersedia untuk mempelajari bahasa query Cypher. Di sini, beberapa contoh cepat kueri OpenCypher dapat membantu Anda mendapatkan gambaran tentang bahasa tersebut, tetapi sejauh ini cara terbaik dan termudah untuk mulai menggunakan OpenCypher untuk menanyakan grafik Neptunus Anda adalah dengan menggunakan notebook OpenCypher di meja kerja Neptunus. Meja kerja adalah sumber terbuka, dan di-host di GitHub . https://github.com/aws-samples/amazon-neptune-samples

Anda akan menemukan notebook OpenCypher di repositori grafik-notebook GitHub Neptunus. Secara khusus, lihat visualisasi rute Udara, dan notebook Tim Premier Inggris untuk OpenCypher.

Data yang diproses oleh OpenCypher mengambil bentuk serangkaian peta yang tidak teratur. key/value Cara utama untuk memperbaiki, memanipulasi, dan menambah peta ini adalah dengan menggunakan klausa yang melakukan tugas-tugas seperti pencocokan pola, penyisipan, pembaruan, dan penghapusan pada pasangan. key/value

Ada beberapa klausa di OpenCypher untuk menemukan pola data dalam grafik, yang `MATCH` paling umum. `MATCH`memungkinkan Anda menentukan pola node, hubungan, dan filter yang ingin Anda cari dalam grafik Anda. Contoh:

- Dapatkan semua node

```
MATCH (n) RETURN n
```

- Temukan node yang terhubung

```
MATCH (n)-[r]->(d) RETURN n, r, d
```

- Temukan jalan

```
MATCH p=(n)-[r]->(d) RETURN p
```

- Dapatkan semua node dengan label

```
MATCH (n:airport) RETURN n
```

Perhatikan bahwa kueri pertama di atas mengembalikan setiap node dalam grafik Anda, dan dua berikutnya mengembalikan setiap node yang memiliki hubungan — ini umumnya tidak disarankan! Di hampir semua kasus, Anda ingin mempersempit data yang dikembalikan, yang dapat Anda lakukan dengan menentukan label dan properti node atau hubungan, seperti pada contoh keempat.

Anda dapat menemukan lembar contekan praktis untuk sintaks OpenCypher di repositori sampel github Neptunus.

Servlet status Neptunus OpenCypher dan titik akhir status

Titik akhir status OpenCypher menyediakan akses ke informasi tentang kueri yang saat ini berjalan di server atau menunggu untuk dijalankan. Ini juga memungkinkan Anda membatalkan kueri tersebut. Titik akhir adalah:

```
https://(the server):(the port number)/openCypher/status
```

Anda dapat menggunakan HTTP `GET` dan `POST` metode untuk mendapatkan status saat ini dari server, atau untuk membatalkan kueri. Anda juga dapat menggunakan `DELETE` metode ini untuk membatalkan kueri yang sedang berjalan atau menunggu.

Parameter untuk permintaan status

Parameter kueri status

- **`includeWaiting`**(`trueataufalse`) — Ketika disetel ke `true` dan parameter lain tidak ada, menyebabkan informasi status untuk permintaan menunggu dikembalikan serta untuk menjalankan kueri.

- **`cancelQuery`**— Digunakan hanya dengan `GET` dan `POST` metode, untuk menunjukkan bahwa ini adalah permintaan pembatalan. `DELETE`Metode ini tidak memerlukan parameter ini.

  Nilai `cancelQuery` parameter tidak digunakan, tetapi ketika `cancelQuery` ada, `queryId` parameter diperlukan, untuk mengidentifikasi kueri mana yang akan dibatalkan.
- **`queryId`**— Berisi ID dari kueri tertentu.

  Ketika digunakan dengan `POST` metode `GET` or dan `cancelQuery` parameter tidak ada, `queryId` menyebabkan informasi status dikembalikan untuk kueri spesifik yang diidentifikasi. Jika `cancelQuery` parameter ada, maka kueri spesifik yang `queryId` mengidentifikasi dibatalkan.

  Saat digunakan dengan `DELETE` metode ini, `queryId` selalu menunjukkan kueri tertentu yang akan dibatalkan.
- **`silent`**— Hanya digunakan saat membatalkan kueri. Jika diatur `ketrue`, menyebabkan pembatalan terjadi secara diam-diam.

Bidang respons permintaan status

Bidang respons status jika ID kueri tertentu tidak disediakan

- acceptedQueryCount— Jumlah kueri yang telah diterima tetapi belum selesai, termasuk kueri dalam antrian.
- runningQueryCount— Jumlah kueri OpenCypher yang sedang berjalan.
- query — Daftar query OpenCypher saat ini.

Bidang respons status untuk kueri tertentu

- queryId — id GUID untuk kueri. Neptune secara otomatis memberikan nilai ID ini ke setiap kueri, atau Anda juga dapat menetapkan ID Anda sendiri (lihat Menyuntikkan ID Kustom Ke Dalam Gremlin Neptune atau Kueri SPARQL).
- queryString — Query yang dikirimkan. Ini dipotong menjadi 1024 karakter jika lebih panjang dari itu.
- queryEvalStats— Statistik untuk kueri ini:
  - menunggu - Menunjukkan berapa lama kueri menunggu, dalam milidetik.
  - elapsed — Jumlah milidetik kueri telah berjalan sejauh ini.
  - dibatalkan — `True` menunjukkan bahwa kueri dibatalkan, atau `False` belum dibatalkan.

Contoh permintaan dan tanggapan status

- Permintaan status semua pertanyaan, termasuk yang menunggu:

```
curl https://server:port/openCypher/status \
  --data-urlencode "includeWaiting=true"
```

  Tanggapan:

```
{
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- Permintaan status kueri yang sedang berjalan, tidak termasuk yang menunggu::

```
curl https://server:port/openCypher/status
```

  Tanggapan:

```
{
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- Permintaan status kueri tunggal:

```
curl https://server:port/openCypher/status \
 --data-urlencode "queryId=eadc6eea-698b-4a2f-8554-5270ab17ebee"
```

  Tanggapan:

```
{
  "queryId" : "eadc6eea-698b-4a2f-8554-5270ab17ebee",
  "queryString" : "MATCH (n1)-[:knows]->(n2), (n2)-[:knows]->(n3), (n3)-[:knows]-
>(n4), (n4)-[:knows]->(n5), (n5)-[:knows]->(n6), (n6)-[:knows]->(n7), (n7)-[:knows]-
>(n8), (n8)-[:knows]->(n9), (n9)-[:knows]->(n10) RETURN COUNT(n1);",
  "queryEvalStats" : {
    "waited" : 0,
```

```
    "elapsed" : 23463,
    "cancelled" : false
  }
}
```

- Permintaan untuk membatalkan kueri

  1. MenggunakanPOST :

```
curl -X POST https://server:port/openCypher/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=f43ce17b-db01-4d37-a074-c76d1c26d7a9"
```

  Tanggapan:

```
{
  "status" : "200 OK",
  "payload" : true
}
```

  2. MenggunakanGET :

```
curl -X GET https://server:port/openCypher/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=588af350-cfde-4222-bee6-b9cedc87180d"
```

  Tanggapan:

```
{
  "status" : "200 OK",
  "payload" : true
}
```

  3. MenggunakanDELETE :

```
curl -X DELETE \
  -s "https://server:port/openCypher/status?queryId=b9a516d1-d25c-4301-
bb80-10b2743ecf0e"
```

  Tanggapan:

```
{
  "status" : "200 OK",
  "payload" : true
}
```

Titik akhir HTTPS Amazon OpenCypher Neptunus

Topik

- OpenCypher membaca dan menulis kueri pada titik akhir HTTPS
- Format hasil OpenCypher JSON default
- Header tambahan HTTP opsional untuk respons multi-bagian OpenCypher

> **Note**
>
> Neptunus saat ini tidak mendukung HTTP/2 untuk permintaan REST API. Klien harus menggunakan HTTP/1.1 saat menghubungkan ke titik akhir.

OpenCypher membaca dan menulis kueri pada titik akhir HTTPS

Titik akhir OpenCypher HTTPS mendukung kueri baca dan pembaruan menggunakan metode `GET` dan metode. `POST` `PUT`Metode `DELETE` dan tidak didukung.

Instruksi berikut memandu Anda melalui koneksi ke OpenCypher titik akhir menggunakan `curl` perintah dan HTTPS. Anda harus mengikuti petunjuk ini dari instans Amazon EC2 di virtual private cloud (VPC) yang sama seperti instans DB Neptune Anda.

Sintaksnya adalah:

```
HTTPS://(the server):(the port number)/openCypher
```

Berikut adalah contoh kueri baca, yang menggunakan `POST` dan yang menggunakan`GET`:

1. Menggunakan`POST` :

```
curl HTTPS://server:port/openCypher \
```

```
  -d "query=MATCH (n1) RETURN n1;"
```

2. Menggunakan GET (string kueri dikodekan URL):

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=MATCH%20(n1)%20RETURN%20n1"
```

Berikut adalah contoh write/update kueri, salah satu yang menggunakan POST dan satu yang menggunakanGET:

1. MenggunakanPOST :

```
curl HTTPS://server:port/openCypher \
  -d "query=CREATE (n:Person { age: 25 })"
```

2. Menggunakan GET (string kueri dikodekan URL):

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=CREATE%20(n%3APerson%20%7B%20age%3A
%2025%20%7D)"
```

Format hasil OpenCypher JSON default

Format JSON berikut dikembalikan secara default, atau dengan mengatur header permintaan secara eksplisit ke. `Accept: application/json` Format ini dirancang agar mudah diurai menjadi objek menggunakan fitur bahasa asli dari sebagian besar perpustakaan.

Dokumen JSON yang dikembalikan berisi satu bidang`results`, yang berisi nilai pengembalian kueri. Contoh di bawah ini menunjukkan pemformatan JSON untuk nilai-nilai umum.

Contoh respons nilai:

```
{
  "results": [
    {
      "count(a)": 121
    }
  ]
}
```

Contoh respons simpul:

```
{
  "results": [
    {
      "a": {
        "~id": "22",
        "~entityType": "node",
        "~labels": [
          "airport"
        ],
        "~properties": {
          "desc": "Seattle-Tacoma",
          "lon": -122.30899810791,
          "runways": 3,
          "type": "airport",
          "country": "US",
          "region": "US-WA",
          "lat": 47.4490013122559,
          "elev": 432,
          "city": "Seattle",
          "icao": "KSEA",
          "code": "SEA",
          "longest": 11901
        }
      }
    }
  ]
}
```


Contoh respons hubungan:


```
{
  "results": [
    {
      "r": {
        "~id": "7389",
        "~entityType": "relationship",
        "~start": "22",
        "~end": "151",
        "~type": "route",
        "~properties": {
          "dist": 956
        }
      }
```

```
    }
  ]
}
```

Contoh respons jalur:

```
{
  "results": [
    {
      "p": [
        {
          "~id": "22",
          "~entityType": "node",
          "~labels": [
            "airport"
          ],
          "~properties": {
            "desc": "Seattle-Tacoma",
            "lon": -122.30899810791,
            "runways": 3,
            "type": "airport",
            "country": "US",
            "region": "US-WA",
            "lat": 47.4490013122559,
            "elev": 432,
            "city": "Seattle",
            "icao": "KSEA",
            "code": "SEA",
            "longest": 11901
          }
        },
        {
          "~id": "7389",
          "~entityType": "relationship",
          "~start": "22",
          "~end": "151",
          "~type": "route",
          "~properties": {
            "dist": 956
          }
        },
        {
          "~id": "151",
```

```
          "~entityType": "node",
          "~labels": [
            "airport"
          ],
          "~properties": {
            "desc": "Ontario International Airport",
            "lon": -117.600997924805,
            "runways": 2,
            "type": "airport",
            "country": "US",
            "region": "US-CA",
            "lat": 34.0559997558594,
            "elev": 944,
            "city": "Ontario",
            "icao": "KONT",
            "code": "ONT",
            "longest": 12198
          }
        }
      ]
    }
  ]
}
```

Header tambahan HTTP opsional untuk respons multi-bagian OpenCypher

Fitur ini tersedia dimulai dengan rilis mesin Neptunus 1.4.5.0.

Respons HTTP terhadap OpenCypher kueri dan pembaruan biasanya dikembalikan dalam beberapa potongan. Ketika kegagalan terjadi setelah potongan respons awal dikirim (dengan kode status HTTP 200), mungkin sulit untuk mendiagnosis masalah. Secara default, `Neptunus melaporkan kegagalan tersebut dengan menambahkan pesan kesalahan ke badan pesan, yang mungkin rusak karena sifat streaming dari respons.

Menggunakan header trailing

Untuk meningkatkan deteksi dan diagnosis kesalahan, Anda dapat mengaktifkan header trailing dengan menyertakan header trailer transfer-encoding (TE) (te: trailer) dalam permintaan Anda. Melakukan hal ini akan menyebabkan Neptune menyertakan dua bidang header baru dalam header trailing dari potongan respons:

- `X-Neptune-Status`— berisi kode respons diikuti dengan nama pendek. Misalnya, dalam kasus keberhasilan, header trailing akan berupa: `X-Neptune-Status: 200 OK`. Dalam kasus kegagalan, kode respons akan menjadi kode kesalahan mesin Neptunus seperti. `X-Neptune-Status: 500 TimeLimitExceededException`
- `X-Neptune-Detail`— kosong untuk permintaan yang berhasil. Dalam kasus kesalahan, ia berisi pesan kesalahan JSON. Karena hanya karakter ASCII yang diperbolehkan dalam nilai header HTTP, string JSON di-enkode dengan URL. Pesan kesalahan juga masih ditambahkan ke badan pesan respons.

Untuk informasi selengkapnya, lihat halaman MDN tentang header permintaan TE.

OpenCypher contoh penggunaan header trailing

Contoh ini menunjukkan bagaimana header trailing membantu mendiagnosis kueri yang melebihi batas waktunya:

```
curl --raw 'https://your-neptune-endpoint:port/openCypher' \
-H 'TE: trailers' \
-d 'query=MATCH(n) RETURN n.firstName'


Output:
< HTTP/1.1 200 OK
< transfer-encoding: chunked
< trailer: X-Neptune-Status, X-Neptune-Detail
< content-type: application/json;charset=UTF-8
<
<
{
  "results": [{
      "n.firstName": "Hossein"
    }, {
      "n.firstName": "Jan"
    }, {
      "n.firstName": "Miguel"
    }, {
      "n.firstName": "Eric"
    },
{"detailedMessage":"Operation terminated (deadline exceeded)",
"code":"TimeLimitExceededException",
"requestId":"a7e9d2aa-fbb7-486e-8447-2ef2a8544080",
```

```
"message":"Operation terminated (deadline exceeded)"}
0
X-Neptune-Status: 500 TimeLimitExceededException
X-Neptune-Detail: %7B%22detailedMessage%22%3A%22Operation+terminated+%28deadline
+exceeded%29%22%2C%22code%22%3A%22TimeLimitExceededException%22%2C%22requestId%22%3A
%22a7e9d2aa-fbb7-486e-8447-2ef2a8544080%22%2C%22message%22%3A%22Operation+terminated+
%28deadline+exceeded%29%22%7D
```

Rincian respons:

Contoh sebelumnya menunjukkan bagaimana OpenCypher respons dengan header tambahan dapat membantu mendiagnosis kegagalan kueri. Di sini kita melihat empat bagian berurutan: (1) header awal dengan status 200 OK yang menunjukkan streaming dimulai, (2) hasil JSON sebagian (rusak) berhasil dialirkan sebelum kegagalan, (3) pesan kesalahan yang ditambahkan yang menunjukkan batas waktu, dan (4) header trailing yang berisi status akhir (500) dan informasi kesalahan terperinci. TimeLimitExceededException

Menggunakan AWS SDK untuk menjalankan kueri OpenCypher

Dengan AWS SDK, Anda dapat menjalankan kueri OpenCypher terhadap grafik Neptunus Anda menggunakan bahasa pemrograman pilihan Anda. Neptunus data API SDK (`neptunedata`nama layanan) menyediakan tindakan untuk mengirimkan kueri ExecuteOpenCypherQueryOpenCypher.

Anda harus menjalankan contoh ini dari instans Amazon EC2 di virtual private cloud (VPC) yang sama dengan cluster DB Neptunus Anda, atau dari lokasi yang memiliki konektivitas jaringan ke titik akhir cluster Anda.

Tautan langsung ke dokumentasi referensi API untuk `neptunedata` layanan di setiap bahasa SDK dapat ditemukan di bawah ini:

Bahasa pemrograman	referensi API neptunedata
C++	https://sdk.amazonaws.com/cpp/api/LATEST/aws-cpp-sdk-neptunedata/html/annotated.html
Go	https://docs.aws.amazon.com/sdk-for-go/api/service/neptunedata/

Bahasa pemrograman	referensi API neptunedata
Java	https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/services/neptunedata/package-summary.html
JavaScript	https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/Package/-aws-sdk-client-neptunedata/
Kotlin	https://sdk.amazonaws.com/kotlin/api/latest/neptunedata/index.html
.NET	https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Neptunedata/NNeptunedata.html
PHP	https://docs.aws.amazon.com/aws-sdk-php/v3/api/namespace-Aws.Neptunedata.html
Python	https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/neptunedata.html
Ruby	https://docs.aws.amazon.com/sdk-for-ruby/v3/api/Aws/Neptunedata.html
Karat	https://crates.io/crates/aws-sdk-neptunedata
CLI	https://docs.aws.amazon.com/cli/latest/reference/neptunedata/

Contoh SDK OpenCypher AWS

Contoh berikut menunjukkan cara menyiapkan `neptunedata` klien, menjalankan kueri OpenCypher, dan mencetak hasilnya. Ganti *`YOUR_NEPTUNE_HOST`* dan *`YOUR_NEPTUNE_PORT`* dengan titik akhir dan port cluster DB Neptunus Anda.

⚠ Konfigurasi batas waktu sisi klien dan coba lagi

Batas waktu klien SDK mengontrol berapa lama klien menunggu respons. Itu tidak mengontrol berapa lama kueri berjalan di server. Jika waktu klien habis sebelum server selesai, kueri dapat terus berjalan di Neptunus sementara klien tidak memiliki cara untuk mengambil hasilnya.

Sebaiknya atur batas waktu baca sisi klien ke `0` (tanpa batas waktu) atau ke nilai yang setidaknya beberapa detik lebih lama dari pengaturan neptune_query_timeout sisi server di cluster DB Neptunus Anda. Ini memungkinkan Neptunus mengontrol saat waktu kueri habis. Kami juga merekomendasikan pengaturan upaya coba ulang maksimum ke 1 (tidak ada percobaan ulang). Jika SDK mencoba ulang kueri yang masih berjalan di server, itu dapat menghasilkan operasi duplikat. Ini sangat penting untuk kueri mutasi, di mana percobaan lagi dapat menyebabkan duplikat penulisan yang tidak diinginkan.

Python

1. 4. Ikuti instruksi instalasi untuk menginstal Boto3.
2. Buat file bernama `openCypherExample.py` dan tempel kode berikut:

```
import boto3
import json
from botocore.config import Config

# Disable the client-side read timeout and retries so that
# Neptune's server-side neptune_query_timeout controls query duration.
client = boto3.client(
    'neptunedata',
    endpoint_url=f'https://YOUR_NEPTUNE_HOST:YOUR_NEPTUNE_PORT',
    config=Config(read_timeout=None, retries={'total_max_attempts': 1})
)

response = client.execute_open_cypher_query(
    openCypherQuery='MATCH (n) RETURN n LIMIT 1'
)

print(json.dumps(response['results'], indent=2))
```

3. Jalankan contoh: `python openCypherExample.py`

Java

1. Ikuti petunjuk penginstalan untuk menyiapkan AWS SDK for Java.
2. Gunakan kode berikut untuk menyiapkan`NeptunedataClient`, menjalankan kueri OpenCypher, dan mencetak hasilnya:

```
import java.net.URI;
import java.time.Duration;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.services.neptunedata.NeptunedataClient;
import
 software.amazon.awssdk.services.neptunedata.model.ExecuteOpenCypherQueryRequest;
import
 software.amazon.awssdk.services.neptunedata.model.ExecuteOpenCypherQueryResponse;

// Disable the client-side timeout and retries so that
// Neptune's server-side neptune_query_timeout controls query duration.
NeptunedataClient client = NeptunedataClient.builder()
    .endpointOverride(URI.create("https://YOUR_NEPTUNE_HOST:YOUR_NEPTUNE_PORT"))
    .overrideConfiguration(ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ZERO)
        .retryPolicy(RetryPolicy.none())
        .build())
    .build();

ExecuteOpenCypherQueryRequest request = ExecuteOpenCypherQueryRequest.builder()
    .openCypherQuery("MATCH (n) RETURN n LIMIT 1")
    .build();

ExecuteOpenCypherQueryResponse response = client.executeOpenCypherQuery(request);

System.out.println(response.results().toString());
```

JavaScript

1. Ikuti petunjuk penginstalan untuk menyiapkan AWS SDK. JavaScript Instal paket klien neptunedata:. `npm install @aws-sdk/client-neptunedata`
2. Buat file bernama `openCypherExample.js` dan tempel kode berikut:

```
import { NeptunedataClient, ExecuteOpenCypherQueryCommand } from "@aws-sdk/client-
neptunedata";
import { NodeHttpHandler } from "@smithy/node-http-handler";

const config = {
    endpoint: "https://YOUR_NEPTUNE_HOST:YOUR_NEPTUNE_PORT",
    // Disable the client-side request timeout so that
    // Neptune's server-side neptune_query_timeout controls query duration.
    requestHandler: new NodeHttpHandler({
        requestTimeout: 0
    }),
    maxAttempts: 1
};

const client = new NeptunedataClient(config);

const input = {
    openCypherQuery: "MATCH (n) RETURN n LIMIT 1"
};

const command = new ExecuteOpenCypherQueryCommand(input);
const response = await client.send(command);

console.log(JSON.stringify(response, null, 2));
```

3. Jalankan contoh: `node openCypherExample.js`

Menggunakan protokol Bolt untuk membuat kueri OpenCypher ke Neptunus

Bolt adalah client/server protokol berorientasi pernyataan yang awalnya dikembangkan oleh Neo4j dan dilisensikan di bawah lisensi Creative Commons 3.0 Attribution-. ShareAlike Ini didorong oleh klien, artinya klien selalu memulai pertukaran pesan.

Untuk terhubung ke Neptunus menggunakan driver Bolt Neo4j, cukup ganti URL dan nomor Port dengan titik akhir cluster Anda menggunakan skema URI. `bolt` Jika Anda memiliki satu instance Neptunus yang berjalan, gunakan endpoint read_write. Jika beberapa instance berjalan, maka dua driver direkomendasikan, satu untuk penulis dan satu lagi untuk semua replika baca. Jika Anda hanya memiliki dua titik akhir default, driver read_write dan read_only sudah cukup, tetapi jika Anda memiliki titik akhir khusus juga, pertimbangkan untuk membuat instance driver untuk masing-masing titik akhir.

> **Note**
>
> Meskipun spesifikasi Bolt menyatakan bahwa Bolt dapat terhubung menggunakan TCP atau, WebSockets Neptunus hanya mendukung koneksi TCP untuk Bolt.

Neptunus memungkinkan hingga 1000 koneksi Bolt bersamaan pada semua ukuran instans kecuali untuk t3.medium dan t4g.medium. Pada instans t3.medium dan t4g.medium hanya 512 koneksi yang diizinkan.

Untuk contoh kueri OpenCypher dalam berbagai bahasa yang menggunakan driver Bolt, lihat dokumentasi Neo4j Drivers & Language Guides.

> **Important**
>
> Driver Neo4j Bolt untuk Python, .NET JavaScript, dan Golang awalnya tidak mendukung pembaruan otomatis token otentikasi Signature v4. AWS Ini berarti bahwa setelah tanda tangan kedaluwarsa (seringkali dalam 5 menit), pengemudi gagal mengautentikasi, dan permintaan berikutnya gagal. Contoh Python, .NET JavaScript, dan Go di bawah ini semuanya terpengaruh oleh masalah ini.
> Lihat masalah driver Python Neo4j #834, masalah Neo4j .NET #664, masalah driverNeo4j #993, dan masalah JavaScriptdriver Neo4j GoLang #429 untuk informasi lebih lanjut.
> Pada driver versi 5.8.0, API otentikasi ulang pratinjau baru dirilis untuk driver Go (lihat v5.8.0 - Umpan balik diinginkan pada otentikasi ulang).

Menggunakan Bolt dengan Java untuk terhubung ke Neptunus

Anda dapat mengunduh driver untuk versi apa pun yang ingin Anda gunakan dari repositori Maven MVN, atau dapat menambahkan ketergantungan ini ke proyek Anda:

```
<dependency>
  <groupId>org.neo4j.driver</groupId>
  <artifactId>neo4j-java-driver</artifactId>
  <version>4.3.3</version>
</dependency>
```

Kemudian, untuk terhubung ke Neptunus di Jawa menggunakan salah satu driver Bolt ini, buat instance driver untuk instance primary/writer di cluster Anda menggunakan kode seperti berikut:

```
import org.neo4j.driver.Driver;
import org.neo4j.driver.GraphDatabase;

final Driver driver =
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
    AuthTokens.none(),
    Config.builder().withEncryption()
                    .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                    .build());
```

Jika Anda memiliki satu atau lebih replika pembaca, Anda juga dapat membuat instance driver untuk mereka menggunakan kode seperti ini:

```
final Driver read_only_driver =              // (without connection timeout)
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
      Config.builder().withEncryption()
                      .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                      .build());
```

Atau, dengan batas waktu:

```
final Driver read_only_timeout_driver =      // (with connection timeout)
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
    Config.builder().withConnectionTimeout(30, TimeUnit.SECONDS)
                    .withEncryption()
                    .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                    .build());
```

Jika Anda memiliki titik akhir khusus, mungkin juga bermanfaat untuk membuat instance driver untuk masing-masing titik.

Contoh kueri Python OpenCypher menggunakan Bolt

Berikut adalah cara membuat query OpenCypher dengan Python menggunakan Bolt:

```
python -m pip install neo4j
```

```
from neo4j import GraphDatabase
uri = "bolt://(your cluster endpoint URL):(your cluster port)"
driver = GraphDatabase.driver(uri, auth=("username", "password"), encrypted=True)
```

Perhatikan bahwa `auth` parameter diabaikan.

Contoh kueri.NET OpenCypher menggunakan Bolt

Untuk membuat kueri OpenCypher di.NET menggunakan Bolt, langkah pertama adalah menginstal driver Neo4j menggunakan. NuHet Untuk melakukan panggilan sinkron, gunakan `.Simple` versi, seperti ini:

```
Install-Package Neo4j.Driver.Simple-4.3.0
```

```
using Neo4j.Driver;

namespace hello
{
  // This example creates a node and reads a node in a Neptune
  // Cluster where IAM Authentication is not enabled.
  public class HelloWorldExample : IDisposable
  {
    private bool _disposed = false;
    private readonly IDriver _driver;
    private static string url = "bolt://(your cluster endpoint URL):(your cluster
 port)";
    private static string createNodeQuery = "CREATE (a:Greeting) SET a.message =
 'HelloWorldExample'";
    private static string readNodeQuery = "MATCH(n:Greeting) RETURN n.message";

    ~HelloWorldExample() => Dispose(false);

    public HelloWorldExample(string uri)
    {
      _driver = GraphDatabase.Driver(uri, AuthTokens.None, o =>
 o.WithEncryptionLevel(EncryptionLevel.Encrypted));
    }

    public void createNode()
    {
      // Open a session
      using (var session = _driver.Session())
      {
         // Run the query in a write transaction
        var greeting = session.WriteTransaction(tx =>
        {
          var result = tx.Run(createNodeQuery);
```

```
          // Consume the result
          return result.Consume();
        });

        // The output will look like this:
        //   ResultSummary{Query=`CREATE (a:Greeting) SET a.message =
'HelloWorldExample".....
        Console.WriteLine(greeting);
      }
    }

    public void retrieveNode()
    {
      // Open a session
      using (var session = _driver.Session())
      {
        // Run the query in a read transaction
        var greeting = session.ReadTransaction(tx =>
        {
          var result = tx.Run(readNodeQuery);
          // Consume the result. Read the single node
          // created in a previous step.
          return result.Single()[0].As<string>();
        });
        // The output will look like this:
        //   HelloWorldExample
        Console.WriteLine(greeting);
      }
    }

    public void Dispose()
    {
      Dispose(true);
      GC.SuppressFinalize(this);
    }

    protected virtual void Dispose(bool disposing)
    {
      if (_disposed)
        return;
      if (disposing)
      {
        _driver?.Dispose();
      }
```

```
      _disposed = true;
    }

    public static void Main()
    {
      using (var apiCaller = new HelloWorldExample(url))
      {
        apiCaller.createNode();
        apiCaller.retrieveNode();
      }
    }
  }
}
```

Contoh kueri Java OpenCypher menggunakan Bolt dengan otentikasi IAM

Kode Java di bawah ini menunjukkan cara membuat query OpenCypher di Java menggunakan Bolt dengan otentikasi IAM. JavaDoc Komentar tersebut menjelaskan penggunaannya. Setelah instance driver tersedia, Anda dapat menggunakannya untuk membuat beberapa permintaan yang diautentikasi.

```
package software.amazon.neptune.bolt;

import com.amazonaws.DefaultRequest;
import com.amazonaws.Request;
import com.amazonaws.auth.AWS4Signer;
import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.http.HttpMethodName;
import com.google.gson.Gson;
import lombok.Builder;
import lombok.Getter;
import lombok.NonNull;
import org.neo4j.driver.Value;
import org.neo4j.driver.Values;
import org.neo4j.driver.internal.security.InternalAuthToken;
import org.neo4j.driver.internal.value.StringValue;

import java.net.URI;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;

import static com.amazonaws.auth.internal.SignerConstants.AUTHORIZATION;
```

```
import static com.amazonaws.auth.internal.SignerConstants.HOST;
import static com.amazonaws.auth.internal.SignerConstants.X_AMZ_DATE;
import static com.amazonaws.auth.internal.SignerConstants.X_AMZ_SECURITY_TOKEN;

/**
 * Use this class instead of `AuthTokens.basic` when working with an IAM
 * auth-enabled server. It works the same as `AuthTokens.basic` when using
 * static credentials, and avoids making requests with an expired signature
 * when using temporary credentials. Internally, it generates a new signature
 * on every invocation (this may change in a future implementation).
 *
 * Note that authentication happens only the first time for a pooled connection.
 *
 * Typical usage:
 *
 * NeptuneAuthToken authToken = NeptuneAuthToken.builder()
 *     .credentialsProvider(credentialsProvider)
 *     .region("aws region")
 *     .url("cluster endpoint url")
 *     .build();
 *
 * Driver driver = GraphDatabase.driver(
 *     authToken.getUrl(),
 *     authToken,
 *     config
 * );
 */

public class NeptuneAuthToken extends InternalAuthToken {
  private static final String SCHEME = "basic";
  private static final String REALM = "realm";
  private static final String SERVICE_NAME = "neptune-db";
  private static final String HTTP_METHOD_HDR = "HttpMethod";
  private static final String DUMMY_USERNAME = "username";
  @NonNull
  private final String region;
  @NonNull
  @Getter
  private final String url;
  @NonNull
  private final AWSCredentialsProvider credentialsProvider;
  private final Gson gson = new Gson();

  @Builder
```

```
  private NeptuneAuthToken(
      @NonNull final String region,
      @NonNull final String url,
      @NonNull final AWSCredentialsProvider credentialsProvider
  ) {
      // The superclass caches the result of toMap(), which we don't want
      super(Collections.emptyMap());
      this.region = region;
      this.url = url;
      this.credentialsProvider = credentialsProvider;
  }

  @Override
  public Map<String, Value> toMap() {
    final Map<String, Value> map = new HashMap<>();
    map.put(SCHEME_KEY, Values.value(SCHEME));
    map.put(PRINCIPAL_KEY, Values.value(DUMMY_USERNAME));
    map.put(CREDENTIALS_KEY, new StringValue(getSignedHeader()));
    map.put(REALM_KEY, Values.value(REALM));

    return map;
  }

  private String getSignedHeader() {
    final Request<Void> request = new DefaultRequest<>(SERVICE_NAME);
    request.setHttpMethod(HttpMethodName.GET);
    request.setEndpoint(URI.create(url));
    // Comment out the following line if you're using an engine version older than
1.2.0.0
    request.setResourcePath("/opencypher");

    final AWS4Signer signer = new AWS4Signer();
    signer.setRegionName(region);
    signer.setServiceName(request.getServiceName());
    signer.sign(request, credentialsProvider.getCredentials());

    return getAuthInfoJson(request);
  }

  private String getAuthInfoJson(final Request<Void> request) {
    final Map<String, Object> obj = new HashMap<>();
    obj.put(AUTHORIZATION, request.getHeaders().get(AUTHORIZATION));
    obj.put(HTTP_METHOD_HDR, request.getHttpMethod());
    obj.put(X_AMZ_DATE, request.getHeaders().get(X_AMZ_DATE));
```

```
    obj.put(HOST, request.getHeaders().get(HOST));
    obj.put(X_AMZ_SECURITY_TOKEN, request.getHeaders().get(X_AMZ_SECURITY_TOKEN));

    return gson.toJson(obj);
  }
}
```

Contoh kueri OpenCypher Python menggunakan Bolt dengan otentikasi IAM

Kelas Python di bawah ini memungkinkan Anda membuat kueri OpenCypher dengan Python menggunakan Bolt dengan otentikasi IAM:

```
import json

from neo4j import Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import Credentials
from botocore.auth import (
  SigV4Auth,
  _host_from_url,
)

SCHEME = "basic"
REALM = "realm"
SERVICE_NAME = "neptune-db"
DUMMY_USERNAME = "username"
HTTP_METHOD_HDR = "HttpMethod"
HTTP_METHOD = "GET"
AUTHORIZATION = "Authorization"
X_AMZ_DATE = "X-Amz-Date"
X_AMZ_SECURITY_TOKEN = "X-Amz-Security-Token"
HOST = "Host"


class NeptuneAuthToken(Auth):
  def __init__(
    self,
    credentials: Credentials,
    region: str,
    url: str,
    **parameters
  ):
```

```
    # Do NOT add "/opencypher" in the line below if you're using an engine version
 older than 1.2.0.0
    request = AWSRequest(method=HTTP_METHOD, url=url + "/opencypher")
    request.headers.add_header("Host", _host_from_url(request.url))
    sigv4 = SigV4Auth(credentials, SERVICE_NAME, region)
    sigv4.add_auth(request)

    auth_obj = {
      hdr: request.headers[hdr]
      for hdr in [AUTHORIZATION, X_AMZ_DATE, X_AMZ_SECURITY_TOKEN, HOST]
    }
    auth_obj[HTTP_METHOD_HDR] = request.method
    creds: str = json.dumps(auth_obj)
    super().__init__(SCHEME, DUMMY_USERNAME, creds, REALM, **parameters)
```

Anda menggunakan kelas ini untuk membuat driver sebagai berikut:

```
  authToken = NeptuneAuthToken(creds, REGION, URL)
  driver = GraphDatabase.driver(URL, auth=authToken, encrypted=True)
```

Contoh Node.js menggunakan otentikasi IAM dan Bolt

Kode Node.js di bawah ini menggunakan AWS SDK untuk JavaScript versi 3 dan ES6 sintaks untuk membuat driver yang mengautentikasi permintaan:

```
import neo4j from "neo4j-driver";
import { HttpRequest }  from "@smithy/protocol-http";
import { defaultProvider } from "@aws-sdk/credential-provider-node";
import { SignatureV4 } from "@smithy/signature-v4";
import crypto from "@aws-crypto/sha256-js";
const { Sha256 } = crypto;
import assert from "node:assert";

const region = "us-west-2";
const serviceName = "neptune-db";
const host = "(your cluster endpoint URL)";
const port = 8182;
const protocol = "bolt";
const hostPort = host + ":" + port;
const url = protocol + "://" + hostPort;
const createQuery = "CREATE (n:Greeting {message: 'Hello'}) RETURN ID(n)";
const readQuery = "MATCH(n:Greeting) WHERE ID(n) = $id RETURN n.message";
```

```
async function signedHeader() {
  const req = new HttpRequest({
    method: "GET",
    protocol: protocol,
    hostname: host,
    port: port,
    // Comment out the following line if you're using an engine version older than
 1.2.0.0
    path: "/opencypher",
    headers: {
      host: hostPort
    }
  });

  const signer = new SignatureV4({
    credentials: defaultProvider(),
    region: region,
    service: serviceName,
    sha256: Sha256
  });

  return signer.sign(req, { unsignableHeaders: new Set(["x-amz-content-sha256"]) })
    .then((signedRequest) => {
      const authInfo = {
        "Authorization": signedRequest.headers["authorization"],
        "HttpMethod": signedRequest.method,
        "X-Amz-Date": signedRequest.headers["x-amz-date"],
        "Host": signedRequest.headers["host"],
        "X-Amz-Security-Token": signedRequest.headers["x-amz-security-token"]
      };
      return JSON.stringify(authInfo);
    });
}

async function createDriver() {
  let authToken = { scheme: "basic", realm: "realm", principal: "username",
 credentials: await signedHeader() };

  return neo4j.driver(url, authToken, {
      encrypted: "ENCRYPTION_ON",
      trust: "TRUST_SYSTEM_CA_SIGNED_CERTIFICATES",
      maxConnectionPoolSize: 1,
      // logging: neo4j.logging.console("debug")
    }
```

```
  );
}

async function unmanagedTxn(driver) {
  const session = driver.session();
  const tx = session.beginTransaction();
  try {
    const created = await tx.run(createQuery);
    const matched = await tx.run(readQuery, { id: created.records[0].get(0) });
    const msg = matched.records[0].get("n.message");
    assert.equal(msg, "Hello");
    await tx.commit();
  } catch (err) {
    // The transaction will be rolled back, now handle the error.
    console.log(err);
  } finally {
    await session.close();
  }
}

const driver = await createDriver();
try {
  await unmanagedTxn(driver);
} catch (err) {
  console.log(err);
} finally {
  await driver.close();
}
```

Contoh kueri.NET OpenCypher menggunakan Bolt dengan otentikasi IAM

Untuk mengaktifkan otentikasi IAM di .NET, Anda perlu menandatangani permintaan saat membuat koneksi. Contoh di bawah ini menunjukkan cara membuat `NeptuneAuthToken` helper untuk menghasilkan token otentikasi:

```
using Amazon.Runtime;
using Amazon.Util;
using Neo4j.Driver;
using System.Security.Cryptography;
using System.Text;
using System.Text.Json;
using System.Web;
```

```
namespace Hello
{
  /*
   * Use this class instead of `AuthTokens.None` when working with an IAM-auth-enabled
 server.
   *
   * Note that authentication happens only the first time for a pooled connection.
   *
   * Typical usage:
   *
   * var authToken = new NeptuneAuthToken(AccessKey, SecretKey,
 Region).GetAuthToken(Host);
   * _driver = GraphDatabase.Driver(Url, authToken, o =>
 o.WithEncryptionLevel(EncryptionLevel.Encrypted));
   */

  public class NeptuneAuthToken
  {
    private const string ServiceName = "neptune-db";
    private const string Scheme = "basic";
    private const string Realm = "realm";
    private const string DummyUserName = "username";
    private const string Algorithm = "AWS4-HMAC-SHA256";
    private const string AWSRequest = "aws4_request";

    private readonly string _accessKey;
    private readonly string _secretKey;
    private readonly string _region;

    private readonly string _emptyPayloadHash;

    private readonly SHA256 _sha256;


    public NeptuneAuthToken(string awsKey = null, string secretKey = null, string
 region = null)
    {
      var awsCredentials = awsKey == null || secretKey == null
        ? FallbackCredentialsFactory.GetCredentials().GetCredentials()
        : null;

      _accessKey = awsKey ?? awsCredentials.AccessKey;
      _secretKey = secretKey ?? awsCredentials.SecretKey;
```

```
      _region = region ?? FallbackRegionFactory.GetRegionEndpoint().SystemName; //ex:
us-east-1

      _sha256 = SHA256.Create();
      _emptyPayloadHash = Hash(Array.Empty<byte>());
    }

    public IAuthToken GetAuthToken(string url)
    {
      return AuthTokens.Custom(DummyUserName, GetCredentials(url), Realm, Scheme);
    }

    /******************** AWS SIGNING FUNCTIONS ********************/
    private string Hash(byte[] bytesToHash)
    {
      return ToHexString(_sha256.ComputeHash(bytesToHash));
    }

    private static byte[] HmacSHA256(byte[] key, string data)
    {
      return new HMACSHA256(key).ComputeHash(Encoding.UTF8.GetBytes(data));
    }

    private byte[] GetSignatureKey(string dateStamp)
    {
      var kSecret = Encoding.UTF8.GetBytes($"AWS4{_secretKey}");
      var kDate = HmacSHA256(kSecret, dateStamp);
      var kRegion = HmacSHA256(kDate, _region);
      var kService = HmacSHA256(kRegion, ServiceName);
      return HmacSHA256(kService, AWSRequest);
    }

    private static string ToHexString(byte[] array)
    {
      return Convert.ToHexString(array).ToLowerInvariant();
    }

    private string GetCredentials(string url)
    {
      var request = new HttpRequestMessage
      {
        Method = HttpMethod.Get,
        RequestUri = new Uri($"https://{url}/opencypher")
      };
```

```
      var signedrequest = Sign(request);

      var headers = new Dictionary<string, object>
      {
        [HeaderKeys.AuthorizationHeader] =
signedrequest.Headers.GetValues(HeaderKeys.AuthorizationHeader).FirstOrDefault(),
        ["HttpMethod"] = HttpMethod.Get.ToString(),
        [HeaderKeys.XAmzDateHeader] =
signedrequest.Headers.GetValues(HeaderKeys.XAmzDateHeader).FirstOrDefault(),
        // Host should be capitalized, not like in Amazon.Util.HeaderKeys.HostHeader
        ["Host"] =
signedrequest.Headers.GetValues(HeaderKeys.HostHeader).FirstOrDefault(),
      };

      return JsonSerializer.Serialize(headers);
    }

    private HttpRequestMessage Sign(HttpRequestMessage request)
    {
      var now = DateTimeOffset.UtcNow;
      var amzdate = now.ToString("yyyyMMddTHHmmssZ");
      var datestamp = now.ToString("yyyyMMdd");

      if (request.Headers.Host == null)
      {
        request.Headers.Host = $"{request.RequestUri.Host}:{request.RequestUri.Port}";
      }

      request.Headers.Add(HeaderKeys.XAmzDateHeader, amzdate);

      var canonicalQueryParams = GetCanonicalQueryParams(request);

      var canonicalRequest = new StringBuilder();
      canonicalRequest.Append(request.Method + "\n");
      canonicalRequest.Append(request.RequestUri.AbsolutePath + "\n");
      canonicalRequest.Append(canonicalQueryParams + "\n");

      var signedHeadersList = new List<string>();
      foreach (var header in request.Headers.OrderBy(a => a.Key.ToLowerInvariant()))
      {
        canonicalRequest.Append(header.Key.ToLowerInvariant());
        canonicalRequest.Append(':');
        canonicalRequest.Append(string.Join(",", header.Value.Select(s => s.Trim())));
```

```
        canonicalRequest.Append('\n');
        signedHeadersList.Add(header.Key.ToLowerInvariant());
      }
      canonicalRequest.Append('\n');

      var signedHeaders = string.Join(";", signedHeadersList);
      canonicalRequest.Append(signedHeaders + "\n");
      canonicalRequest.Append(_emptyPayloadHash);

      var credentialScope = $"{datestamp}/{_region}/{ServiceName}/{AWSRequest}";
      var stringToSign = $"{Algorithm}\n{amzdate}\n{credentialScope}\n"
        + Hash(Encoding.UTF8.GetBytes(canonicalRequest.ToString()));

      var signing_key = GetSignatureKey(datestamp);
      var signature = ToHexString(HmacSHA256(signing_key, stringToSign));

      request.Headers.TryAddWithoutValidation(HeaderKeys.AuthorizationHeader,
        $"{Algorithm} Credential={_accessKey}/{credentialScope},
 SignedHeaders={signedHeaders}, Signature={signature}");

      return request;
    }

    private static string GetCanonicalQueryParams(HttpRequestMessage request)
    {
      var querystring = HttpUtility.ParseQueryString(request.RequestUri.Query);

      // Query params must be escaped in upper case (i.e. "%2C", not "%2c").
      var queryParams = querystring.AllKeys.OrderBy(a => a)
        .Select(key => $"{key}={Uri.EscapeDataString(querystring[key])}");
      return string.Join("&", queryParams);
    }
  }
}
```

Berikut adalah cara membuat query OpenCypher di.NET menggunakan Bolt dengan otentikasi IAM. Contoh di bawah ini menggunakan `NeptuneAuthToken` helper:

```
using Neo4j.Driver;

namespace Hello
{
  public class HelloWorldExample
```

```
 {
   private const string Host = "(your hostname):8182";
   private const string Url = $"bolt://{Host}";
   private const string CreateNodeQuery = "CREATE (a:Greeting) SET a.message =
'HelloWorldExample'";
   private const string ReadNodeQuery = "MATCH(n:Greeting) RETURN n.message";

   private const string AccessKey = "(your access key)";
   private const string SecretKey = "(your secret key)";
   private const string Region = "(your AWS region)"; // e.g. "us-west-2"

   private readonly IDriver _driver;

   public HelloWorldExample()
   {
     var authToken = new NeptuneAuthToken(AccessKey, SecretKey,
Region).GetAuthToken(Host);

     // Note that when the connection is reinitialized after max connection lifetime
     // has been reached, the signature token could have already been expired (usually
5 min)
     // You can face exceptions like:
     //   `Unexpected server exception 'Signature expired: XXXX is now earlier than
YYYY (ZZZZ - 5 min.)`
     _driver = GraphDatabase.Driver(Url, authToken, o =>

o.WithMaxConnectionLifetime(TimeSpan.FromMinutes(60)).WithEncryptionLevel(EncryptionLevel.Encr
   }

   public async Task CreateNode()
   {
     // Open a session
     using (var session = _driver.AsyncSession())
     {
       // Run the query in a write transaction
       var greeting = await session.WriteTransactionAsync(async tx =>
       {
         var result = await tx.RunAsync(CreateNodeQuery);
         // Consume the result
         return await result.ConsumeAsync();
       });

       // The output will look like this:
```

```
        //   ResultSummary{Query=`CREATE (a:Greeting) SET a.message =
 'HelloWorldExample".....
        Console.WriteLine(greeting.Query);
      }
    }

    public async Task RetrieveNode()
    {
      // Open a session
      using (var session = _driver.AsyncSession())
      {
        // Run the query in a read transaction
        var greeting = await session.ReadTransactionAsync(async tx =>
        {
          var result = await tx.RunAsync(ReadNodeQuery);
          var records = await result.ToListAsync();

          // Consume the result. Read the single node
          // created in a previous step.
          return records[0].Values.First().Value;
        });
        // The output will look like this:
        //   HelloWorldExample
        Console.WriteLine(greeting);
      }
    }
  }
}
```

Contoh ini dapat diluncurkan dengan menjalankan kode di bawah ini pada `.NET 6` atau `.NET 7` dengan paket-paket berikut:

- **Neo4j**.Driver=4.3.0
- **AWSSDK**.Core=3.7.102.1

```
namespace Hello
{
  class Program
  {
    static async Task Main()
    {
      var apiCaller = new HelloWorldExample();
```

```
      await apiCaller.CreateNode();
      await apiCaller.RetrieveNode();
    }
  }
}
```

Contoh kueri Golang OpenCypher menggunakan Bolt dengan otentikasi IAM

Paket Golang di bawah ini menunjukkan cara membuat kueri OpenCypher dalam bahasa Go menggunakan Bolt dengan otentikasi IAM:

```
package main

import (
  "context"
  "encoding/json"
  "fmt"
  "github.com/aws/aws-sdk-go/aws/credentials"
  "github.com/aws/aws-sdk-go/aws/signer/v4"
  "github.com/neo4j/neo4j-go-driver/v5/neo4j"
  "log"
  "net/http"
  "os"
  "time"
)

const (
  ServiceName   = "neptune-db"
  DummyUsername = "username"
)

// Find node by id using Go driver
func findNode(ctx context.Context, region string, hostAndPort string, nodeId string)
 (string, error) {
  req, err := http.NewRequest(http.MethodGet, "https://"+hostAndPort+"/opencypher",
 nil)

  if err != nil {
    return "", fmt.Errorf("error creating request, %v", err)
  }

  // credentials must have been exported as environment variables
```

```
  signer := v4.NewSigner(credentials.NewEnvCredentials())
  _, err = signer.Sign(req, nil, ServiceName, region, time.Now())

  if err != nil {
    return "", fmt.Errorf("error signing request: %v", err)
  }

  hdrs := []string{"Authorization", "X-Amz-Date", "X-Amz-Security-Token"}
  hdrMap := make(map[string]string)
  for _, h := range hdrs {
    hdrMap[h] = req.Header.Get(h)
  }

  hdrMap["Host"] = req.Host
  hdrMap["HttpMethod"] = req.Method

  password, err := json.Marshal(hdrMap)
  if err != nil {
    return "", fmt.Errorf("error creating JSON, %v", err)
  }
  authToken := neo4j.BasicAuth(DummyUsername, string(password), "")
  // +s enables encryption with a full certificate check
  // Use +ssc to disable client side TLS verification
  driver, err := neo4j.NewDriverWithContext("bolt+s://"+hostAndPort+"/opencypher",
authToken)
  if err != nil {
    return "", fmt.Errorf("error creating driver, %v", err)
  }

  defer driver.Close(ctx)

  if err := driver.VerifyConnectivity(ctx); err != nil {
    log.Fatalf("failed to verify connection, %v", err)
  }

  config := neo4j.SessionConfig{}

  session := driver.NewSession(ctx, config)
  defer session.Close(ctx)

  result, err := session.Run(
    ctx,
    fmt.Sprintf("MATCH (n) WHERE ID(n) = '%s' RETURN n", nodeId),
    map[string]any{},
```

```
  )
  if err != nil {
    return "", fmt.Errorf("error running query, %v", err)
  }

  if !result.Next(ctx) {
    return "", fmt.Errorf("node not found")
  }

  n, found := result.Record().Get("n")
  if !found {
    return "", fmt.Errorf("node not found")
  }

  return fmt.Sprintf("+%v\n", n), nil
}

func main() {
  if len(os.Args) < 3 {
    log.Fatal("Usage: go main.go (region) (host and port)")
  }
  region := os.Args[1]
  hostAndPort := os.Args[2]
  ctx := context.Background()

  res, err := findNode(ctx, region, hostAndPort,
 "72c2e8c1-7d5f-5f30-10ca-9d2bb8c4afbc")
  if err != nil {
    log.Fatal(err)
  }
  fmt.Println(res)
}
```

Perilaku koneksi baut di Neptunus

Berikut adalah beberapa hal yang perlu diingat tentang koneksi Neptunus Bolt:

- Karena koneksi Bolt dibuat pada layer TCP, Anda tidak dapat menggunakan Application Load Balancer di depannya, seperti yang Anda bisa dengan endpoint HTTP.
- Port yang digunakan Neptunus untuk koneksi Bolt adalah port cluster DB Anda.
- Berdasarkan pembukaan Bolt yang diteruskan ke sana, server Neptunus memilih versi Bolt tertinggi yang sesuai (1, 2, 3, atau 4.0).

- Jumlah maksimum koneksi ke server Neptunus yang dapat dibuka klien kapan saja adalah 1.000.
- Jika klien tidak menutup koneksi setelah kueri, koneksi itu dapat digunakan untuk menjalankan kueri berikutnya.
- Namun, jika koneksi menganggur selama 20 menit, server menutupnya secara otomatis.
- Jika autentikasi IAM tidak diaktifkan, Anda dapat menggunakan `AuthTokens.none()` daripada memasok nama pengguna dan kata sandi dummy. Misalnya, di Jawa:

```
GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
 AuthTokens.none(),

 Config.builder().withEncryption().withTrustStrategy(TrustStrategy.trustSystemCertificates())
```

- Ketika autentikasi IAM diaktifkan, koneksi Bolt selalu terputus beberapa menit lebih dari 10 hari setelah dibuat jika belum ditutup karena alasan lain.
- Jika klien mengirim kueri untuk dieksekusi melalui koneksi tanpa menghabiskan hasil kueri sebelumnya, kueri baru akan dibuang. Untuk membuang hasil sebelumnya, klien harus mengirim pesan reset melalui koneksi.
- Hanya satu transaksi pada satu waktu yang dapat dibuat pada koneksi tertentu.
- Jika pengecualian terjadi selama transaksi, server Neptunus memutar kembali transaksi dan menutup koneksi. Dalam hal ini, driver membuat koneksi baru untuk kueri berikutnya.
- Ketahuilah bahwa sesi tidak aman untuk utas. Beberapa operasi parallel harus menggunakan beberapa sesi terpisah.

Contoh kueri parameter OpenCypher

Neptunus mendukung kueri OpenCypher berparameter. Ini memungkinkan Anda menggunakan struktur kueri yang sama beberapa kali dengan argumen yang berbeda. Karena struktur kueri tidak berubah, Neptunus dapat men-cache pohon sintaks abstrak (AST) daripada harus menguraikannya beberapa kali.

Contoh kueri parameter OpenCypher menggunakan titik akhir HTTPS

Di bawah ini adalah contoh penggunaan kueri parameter dengan titik akhir Neptunus OpenCypher HTTPS. Kuerinya adalah:

```
MATCH (n {name: $name, age: $age})
RETURN n
```

Parameter didefinisikan sebagai berikut:

```
parameters={"name": "john", "age": 20}
```

Dengan menggunakan`GET`, Anda dapat mengirimkan kueri berparameter seperti ini:

```
curl -k \
  "https://localhost:8182/openCypher?query=MATCH%20%28n%20%7Bname:\$name,age:\$age%7D
%29%20RETURN%20n&parameters=%7B%22name%22:%22john%22,%22age%22:20%7D"
```

Atau, Anda dapat menggunakan`POST`:

```
curl -k \
  https://localhost:8182/openCypher \
  -d "query=MATCH (n {name: \$name, age: \$age}) RETURN n" \
  -d "parameters={\"name\": \"john\", \"age\": 20}"
```

Atau, menggunakan`DIRECT POST`:

```
curl -k \
   -H "Content-Type: application/opencypher" \
  "https://localhost:8182/openCypher?parameters=%7B%22name%22:%22john%22,%22age
%22:20%7D" \
  -d "MATCH (n {name: \$name, age: \$age}) RETURN n"
```

Contoh kueri parameter OpenCypher menggunakan Bolt

Berikut adalah contoh Python dari kueri parameter OpenCypher menggunakan protokol Bolt:

```
from neo4j import GraphDatabase
uri = "bolt://[neptune-endpoint-url]:8182"
driver = GraphDatabase.driver(uri, auth=("", ""))

def match_name_and_age(tx, name, age):
  # Parameterized Query
  tx.run("MATCH (n {name: $name, age: $age}) RETURN n", name=name, age=age)

with driver.session() as session:
  # Parameters
  session.read_transaction(match_name_and_age, "john", 20)
```

```
driver.close()
```

Berikut adalah contoh Java dari kueri parameter OpenCypher menggunakan protokol Bolt:

```
Driver driver = GraphDatabase.driver("bolt+s://(your cluster endpoint URL):8182");
HashMap<String, Object> parameters = new HashMap<>();
parameters.put("name", "john");
parameters.put("age", 20);
String queryString = "MATCH (n {name: $name, age: $age}) RETURN n";
Result result = driver.session().run(queryString, parameters);
```

Model data OpenCypher

Mesin Neptunus OpenCypher dibangun di atas model grafik properti yang sama dengan Gremlin. Khususnya:

- Setiap node memiliki satu atau lebih label. Jika Anda menyisipkan node tanpa label, label default bernama `vertex` dilampirkan. Jika Anda mencoba menghapus semua label node, kesalahan akan dilemparkan.
- Hubungan adalah entitas yang memiliki persis satu jenis hubungan dan yang membentuk koneksi searah antara dua node (yaitu, dari salah satu node ke node lainnya).
- Baik node dan hubungan dapat memiliki properti, tetapi tidak harus. Neptunus mendukung node dan hubungan dengan properti nol.
- Neptunus tidak mendukung metaproperties, yang juga tidak termasuk dalam spesifikasi OpenCypher.
- Properti dalam grafik Anda dapat bernilai banyak jika dibuat menggunakan Gremlin. Itu adalah node atau properti hubungan dapat memiliki satu set nilai yang berbeda daripada hanya satu. Neptunus telah memperluas semantik OpenCypher untuk menangani properti multi-nilai dengan anggun.

Tipe data yang didukung didokumentasikan diFormat data OpenCypher. Namun, kami tidak menyarankan memasukkan nilai `Array` properti ke dalam grafik OpenCypher saat ini. Meskipun dimungkinkan untuk memasukkan nilai properti array menggunakan pemuat massal, rilis Neptunus OpenCypher saat ini memperlakukannya sebagai satu set properti multi-nilai alih-alih sebagai nilai daftar tunggal.

Di bawah ini adalah daftar tipe data yang didukung dalam rilis ini:

- `Bool`
- `Byte`
- `Short`
- `Int`
- `Long`
- `Float`(Termasuk plus dan minus Infinity dan NaN, tapi bukan INF)
- `Double`(Termasuk plus dan minus Infinity dan NaN, tapi bukan INF)
- `DateTime`
- `String`

Fitur OpenCypher **`explain`**

`explain`Fitur OpenCypher adalah alat swalayan di Amazon Neptunus yang membantu Anda memahami pendekatan eksekusi yang diambil oleh mesin Neptunus. Untuk memanggil menjelaskan, Anda meneruskan parameter ke permintaan HTTPS OpenCypher dengan`explain=`*`mode`*, di mana *`mode`* nilainya dapat berupa salah satu dari yang berikut:

- **`static`**— Dalam `static` mode, hanya `explain` mencetak struktur statis dari rencana kueri. Itu tidak benar-benar menjalankan kueri.
- **`dynamic`**— Dalam `dynamic` mode, `explain` juga menjalankan kueri, dan mencakup aspek dinamis dari rencana kueri. Ini mungkin termasuk jumlah binding perantara yang mengalir melalui operator, rasio binding masuk ke binding keluar, dan total waktu yang dibutuhkan oleh masing-masing operator.
- **`details`**— Dalam `details` mode, `explain` mencetak informasi yang ditampilkan dalam mode dinamis ditambah detail tambahan, seperti string kueri OpenCypher yang sebenarnya dan perkiraan jumlah rentang untuk pola yang mendasari operator gabungan.

Misalnya, menggunakan`POST`:

```
curl HTTPS://server:port/openCypher \
  -d "query=MATCH (n) RETURN n LIMIT 1;" \
  -d "explain=dynamic"
```

Atau, menggunakan`GET`:

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=MATCH%20(n)%20RETURN%20n%20LIMIT
%201&explain=dynamic"
```

Keterbatasan untuk **explain** OpenCypher di Neptunus

Rilis OpenCypher menjelaskan saat ini memiliki batasan berikut:

- Jelaskan rencana saat ini hanya tersedia untuk kueri yang melakukan operasi hanya-baca. Kueri yang melakukan mutasi apa pun, sepertiCREATE,, DELETEMERGE, SET dan sebagainya, tidak didukung.
- Operator dan output untuk rencana tertentu dapat berubah dalam rilis future.

Operator DFE dalam keluaran OpenCypher **explain**

Untuk menggunakan informasi yang disediakan oleh explain fitur OpenCypher, Anda perlu memahami beberapa detail tentang cara kerja mesin kueri DFE (DFE menjadi mesin yang digunakan Neptunus untuk memproses kueri OpenCypher).

Mesin DFE menerjemahkan setiap kueri ke dalam saluran operator. Mulai dari operator pertama, solusi perantara mengalir dari satu operator ke operator berikutnya melalui pipeline operator ini. Setiap baris dalam tabel jelaskan mewakili hasil, hingga titik evaluasi.

Operator yang dapat muncul dalam paket kueri DFE adalah sebagai berikut:

DFEApply— Mengeksekusi fungsi yang ditentukan di bagian argumen, pada nilai yang disimpan dalam variabel tertentu

DFEBindRelasi - Mengikat variabel dengan nama yang ditentukan

DFEChunkLocalSubQuery— Ini adalah operasi non-pemblokiran yang bertindak sebagai pembungkus di sekitar subquery yang sedang dilakukan.

DFEDistinctKolom - Mengembalikan subset yang berbeda dari nilai masukan berdasarkan variabel yang ditentukan.

DFEDistinctRelasi — Mengembalikan subset yang berbeda dari solusi masukan berdasarkan variabel yang ditentukan.

DFEDrain— Muncul di akhir subquery untuk bertindak sebagai langkah penghentian untuk subquery itu. Jumlah solusi dicatat sebagaiUnits In. Units Outselalu nol.

DFEForwardNilai - Menyalin semua potongan input secara langsung sebagai potongan keluaran untuk diteruskan ke operator hilirnya.

DFEGroupByHashIndex— Melakukan operasi kelompok-demi melalui solusi input berdasarkan indeks hash yang dihitung sebelumnya (menggunakan operasi). `DFEHashIndexBuild` Sebagai output, input yang diberikan diperpanjang oleh kolom yang berisi kunci grup untuk setiap solusi masukan.

DFEHashIndexBuild— Membangun indeks hash di atas satu set variabel sebagai efek samping. Indeks hash ini biasanya digunakan kembali dalam operasi selanjutnya. Lihat `DFEHashIndexJoin` atau `DFEGroupByHashIndex` di mana indeks hash ini dapat digunakan.

DFEHashIndexJoin— Melakukan gabungan atas solusi yang masuk terhadap indeks hash yang dibangun sebelumnya. Lihat `DFEHashIndexBuild` di mana indeks hash ini mungkin dibangun.

DFEJoinAda - Mengambil relasi input tangan kiri dan kanan, dan mempertahankan nilai dari relasi kiri yang memiliki nilai yang sesuai dalam relasi kanan seperti yang didefinisikan oleh variabel gabungan yang diberikan.

— Ini adalah operasi non-pemblokiran yang bertindak sebagai pembungkus untuk subquery, yang memungkinkannya dijalankan berulang kali untuk digunakan dalam loop.

DFEMergePotongan — Ini adalah operasi pemblokiran yang menggabungkan potongan dari operator hulu menjadi satu potongan solusi untuk diteruskan ke operator hilirnya (kebalikan dari). `DFESplitChunks`

DFEMinus— Mengambil relasi input tangan kiri dan kanan, dan mempertahankan nilai dari relasi kiri yang tidak memiliki nilai yang sesuai dalam relasi kanan seperti yang didefinisikan oleh variabel gabungan yang diberikan. Jika tidak ada tumpang tindih dalam variabel di kedua relasi, maka operator ini hanya mengembalikan relasi input tangan kiri.

DFENotAda - Mengambil relasi input tangan kiri dan kanan, dan mempertahankan nilai dari relasi kiri yang tidak memiliki nilai yang sesuai dalam relasi kanan seperti yang didefinisikan oleh variabel gabungan yang diberikan. Jika tidak ada tumpang tindih dalam variabel di kedua relasi, maka operator ini mengembalikan relasi kosong.

DFEOptionalGabung — Melakukan gabungan luar kiri (juga disebut gabungan OPSIONAL): solusi dari sisi kiri yang memiliki setidaknya satu mitra bergabung di sisi kanan digabungkan, dan solusi dari sisi kiri tanpa mitra bergabung di sisi kanan diteruskan apa adanya. Ini adalah operasi pemblokiran.

DFEPipelineGabung - Bergabung dengan input terhadap pola Tuple yang ditentukan oleh argumen. `pattern`

DFEPipelineRangeCount— Menghitung jumlah solusi yang cocok dengan pola yang diberikan, dan mengembalikan solusi satu-ary tunggal yang berisi nilai hitungan.

DFEPipelineScan — Memindai database untuk `pattern` argumen yang diberikan, dengan atau tanpa filter yang diberikan pada kolom (s).

DFEProject— Mengambil beberapa kolom input dan proyek hanya kolom yang diinginkan.

DFEReduce— Melakukan fungsi agregasi tertentu pada variabel tertentu.

DFERelationalGabung - Bergabung dengan masukan dari operator sebelumnya berdasarkan tombol pola yang ditentukan menggunakan gabungan gabungan. Ini adalah operasi pemblokiran.

DFERoutePotongan - Mengambil potongan input dari tepi masuk tunggal dan merutekan potongan tersebut di sepanjang beberapa tepi keluarnya.

DFESelectBaris — Operator ini secara selektif mengambil baris dari solusi relasi input kirinya untuk diteruskan ke operator hilirnya. Baris yang dipilih berdasarkan pengidentifikasi baris yang disediakan dalam relasi input kanan operator.

DFESerialize— Serialisasi hasil akhir kueri menjadi serialisasi string JSON, memetakan setiap solusi input ke nama variabel yang sesuai. Untuk hasil node dan edge, hasil ini diserialisasikan ke dalam peta properti entitas dan metadata.

DFESort— Mengambil relasi input dan menghasilkan relasi yang diurutkan berdasarkan kunci sortir yang disediakan.

DFESplitByGroup— Membagi setiap potongan input tunggal dari satu tepi masuk menjadi potongan keluaran yang lebih kecil yang sesuai dengan grup baris yang diidentifikasi oleh baris IDs dari potongan input yang sesuai dari tepi masuk lainnya.

DFESplitPotongan — Membagi setiap potongan input tunggal menjadi potongan keluaran yang lebih kecil (kebalikan dari). `DFEMergeChunks`

DFEStreamingHashIndexBuild— Versi streaming dari`DFEHashIndexBuild`.

DFEStreamingGroupByHashIndex— Versi streaming dari`DFEGroupByHashIndex`.

DFESubquery— Operator ini muncul di awal semua rencana dan merangkum bagian-bagian dari rencana yang dijalankan pada mesin DFE, yang merupakan keseluruhan rencana untuk OpenCypher.

DFESymmetricHashJoin— Bergabung dengan masukan dari operator sebelumnya berdasarkan tombol pola yang ditentukan menggunakan gabungan hash. Ini adalah operasi non-pemblokiran.

DFESync— Operator ini adalah operator sinkronisasi yang mendukung rencana non-pemblokiran. Dibutuhkan solusi dari dua sisi yang masuk dan meneruskan solusi ini ke tepi hilir yang sesuai. Untuk tujuan sinkronisasi, input di sepanjang salah satu tepi ini dapat disangga secara internal.

DFETee— Ini adalah operator percabangan yang mengirimkan serangkaian solusi yang sama ke beberapa operator.

DFETermResolusi — Melakukan operasi lokalisasi atau globalisasi pada inputnya, menghasilkan kolom pengidentifikasi lokal atau global masing-masing.

— Membuka daftar nilai dari kolom input ke kolom output sebagai elemen individual.

DFEUnion— Mengambil dua atau lebih hubungan masukan dan menghasilkan penyatuan relasi-relasi tersebut menggunakan skema output yang diinginkan.

SolutionInjection— Muncul sebelum segala sesuatu yang lain dalam output menjelaskan, dengan nilai 1 di kolom Units Out. Namun, ini berfungsi sebagai no-op, dan tidak benar-benar menyuntikkan solusi apa pun ke mesin DFE.

TermResolution— Muncul di akhir rencana dan menerjemahkan objek dari mesin Neptunus ke objek OpenCypher.

Kolom dalam keluaran OpenCypher **`explain`**

Informasi rencana kueri yang dihasilkan Neptunus sebagai OpenCypher menjelaskan output berisi tabel dengan satu operator per baris. Tabel memiliki kolom berikut:

ID — ID numerik operator ini dalam paket.

Keluar #1 (dan Keluar #2) — ID operator yang berada di hilir dari operator ini. Paling banyak ada dua operator hilir.

Nama — Nama operator ini.

Argumen — Setiap detail yang relevan untuk operator. Ini termasuk hal-hal seperti skema input, skema output, pola (untuk `PipelineScan` dan`PipelineJoin`), dan sebagainya.

Mode — Label yang menjelaskan perilaku operator mendasar. Kolom ini sebagian besar kosong (`-`). Satu pengecualian adalah`TermResolution`, di mana mode bisa`id2value_opencypher`, menunjukkan resolusi dari ID ke nilai OpenCypher.

Unit In — Jumlah solusi yang diteruskan sebagai masukan ke operator ini. Operator tanpa operator hulu, seperti`DFEPipelineScan`,`SolutionInjections`, dan tanpa nilai statis `DFESubquery` yang disuntikkan, akan memiliki nilai nol.

Unit Keluar — Jumlah solusi yang dihasilkan sebagai output dari operator ini. `DFEDrain`adalah kasus khusus, di mana jumlah larutan yang dikeringkan dicatat `Units In` dan `Units Out` selalu nol.

Rasio — Rasio `Units Out` terhadap`Units In`.

Waktu (ms) — Waktu CPU yang dikonsumsi oleh operator ini, dalam milidetik.

Contoh dasar OpenCypher menjelaskan output

Berikut ini adalah contoh dasar dari output OpenCypher`explain`. Kueri adalah pencarian simpul tunggal dalam dataset rute udara untuk node dengan kode bandara `ATL` yang memanggil `explain` menggunakan `details` mode dalam format output ASCII default:

```
curl -d "query=MATCH (n {code: 'ATL'}) RETURN n" -k https://localhost:8182/openCypher -
d "explain=details"
                                    ~
Query:
MATCH (n {code: 'ATL'}) RETURN n

#########################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
#########################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 1         # 0.00  # 4.00      #
#########################################################################################
# 2  # -      # -      # TermResolution    # vars=[?n]          # id2value_opencypher #
 1        # 1         # 1.00  # 2.00      #
#########################################################################################
```

```
subQuery1
#########################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                        # Mode # Units
 In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?n) with property 'code'
 as ?n_code2 and label 'ALL'                                             # -    # 0
   # 1         # 0.00  # 0.21      #
#    #        #        #                       # inlineFilters=[(?n_code2 IN
 ["ATL"^^xsd:string])]                                                          #
  #          #           #       #           #
#    #        #        #                       # patternEstimate=1
                                                                       #      #
  #           #       #           #
#########################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#9d84f97c-c3b0-459a-98d5-955a8726b159/graph_1 # -    #
 1        # 1         # 1.00  # 0.04      #
#########################################################################################
# 2  # 3      # -      # DFEProject            # columns=[?n]
                                                                        # -    # 1
  # 1         # 1.00  # 0.04      #
#########################################################################################
# 3  # -      # -      # DFEDrain              # -
                                                                        # -    # 1
  # 0         # 0.00  # 0.03      #
#########################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#9d84f97c-
c3b0-459a-98d5-955a8726b159/graph_1
#########################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                    # Mode # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?n, ?n_code2]
                    # -    # 0        # 1         # 0.00  # 0.02      #
#########################################################################################
# 1  # 2      # 3      # DFETee               # -
                    # -    # 1        # 2         # 2.00  # 0.02      #
#########################################################################################
# 2  # 4      # -      # DFEDistinctColumn    # column=?n
                    # -    # 1        # 1         # 1.00  # 0.20      #
```

```
#    #        #        #                    # ordered=false
                    #      #          #           #       #           #
#########################################################################################
# 3  # 5      # -      # DFEHashIndexBuild    # vars=[?n]
                    # -    # 1        # 1         # 1.00  # 0.04      #
#########################################################################################
# 4  # 5      # -      # DFEPipelineJoin      # pattern=Node(?n) with property 'ALL'
 and label '?n_label1' # -    # 1        # 1         # 1.00  # 0.25      #
#    #        #        #                    # patternEstimate=3506
                    #      #          #           #       #           #
#########################################################################################
# 5  # 6      # 7      # DFESync              # -
                    # -    # 2        # 2         # 1.00  # 0.02      #
#########################################################################################
# 6  # 8      # -      # DFEForwardValue      # -
                    # -    # 1        # 1         # 1.00  # 0.01      #
#########################################################################################
# 7  # 8      # -      # DFEForwardValue      # -
                    # -    # 1        # 1         # 1.00  # 0.01      #
#########################################################################################
# 8  # 9      # -      # DFEHashIndexJoin     # -
                    # -    # 2        # 1         # 0.50  # 0.35      #
#########################################################################################
# 9  # -      # -      # DFEDrain             # -
                    # -    # 1        # 0         # 0.00  # 0.02      #
#########################################################################################
```

Di tingkat atas, `SolutionInjection` muncul sebelum yang lainnya, dengan 1 unit keluar. Perhatikan bahwa itu tidak benar-benar menyuntikkan solusi apa pun. Anda dapat melihat bahwa operator berikutnya,`DFESubquery`, memiliki 0 unit.

Setelah `SolutionInjection` di tingkat atas adalah `DFESubquery` dan `TermResolution` operator. `DFESubquery`merangkum bagian-bagian dari rencana eksekusi kueri yang sedang didorong ke mesin DFE (untuk kueri OpenCypher, seluruh rencana kueri dijalankan oleh DFE). Semua operator dalam paket kueri bersarang di dalam `subQuery1` yang direferensikan oleh. `DFESubquery` Satu-satunya pengecualian adalah`TermResolution`, yang terwujud internal IDs menjadi objek OpenCypher yang sepenuhnya diserialisasi.

Semua operator yang didorong ke mesin DFE memiliki nama yang dimulai dengan `DFE` awalan. Seperti disebutkan di atas, seluruh rencana kueri OpenCypher dijalankan oleh DFE, sehingga sebagai hasilnya, semua operator kecuali operator akhir `TermResolution` memulai dengan. `DFE`

Di dalamsubQuery1, mungkin ada nol atau lebih DFEChunkLocalSubQuery atau DFELoopSubQuery operator yang merangkum bagian dari rencana eksekusi yang didorong yang dijalankan dalam mekanisme yang dibatasi memori. DFEChunkLocalSubQuerydi sini berisi salah satu SolutionInjection yang digunakan sebagai masukan ke subquery. Untuk menemukan tabel untuk subquery tersebut di output, cari yang subQuery=*graph URI* ditentukan di Arguments kolom untuk DFELoopSubQuery operator DFEChunkLocalSubQuery or.

DalamsubQuery1, DFEPipelineScan dengan ID 0 memindai database untuk yang ditentukanpattern. Pola memindai entitas dengan properti yang code disimpan sebagai variabel ?n_code2 di atas semua label (Anda dapat memfilter pada label tertentu dengan menambahkan airport ken:airport). inlineFiltersArgumen menunjukkan penyaringan untuk code properti yang samaATL.

Selanjutnya, DFEChunkLocalSubQuery operator bergabung dengan hasil perantara dari subquery yang berisi. DFEPipelineJoin Ini memastikan bahwa sebenarnya ?n adalah simpul, karena sebelumnya DFEPipelineScan memindai entitas apa pun dengan code properti.

Contoh **explain** output untuk pencarian hubungan dengan batas

Kueri ini mencari hubungan antara dua node anonim dengan tiperoute, dan mengembalikan paling banyak 10. Sekali lagi, explain modenya details dan format output adalah format ASCII default. Berikut adalah explain outputnya:

Di sini, DFEPipelineScan memindai tepi yang dimulai dari simpul anonim ?anon_node7 dan berakhir di simpul anonim lainnya?anon_node21, dengan tipe hubungan disimpan sebagai? p_type1. Ada filter untuk ?p_type1 being el://route (di mana el singkatan dari label tepi), yang sesuai dengan [p:route] dalam string kueri.

DFEDrainmengumpulkan solusi output dengan batas 10, seperti yang ditunjukkan pada Arguments kolomnya. DFEDrainberakhir setelah batas tercapai atau semua solusi diproduksi, mana yang terjadi lebih dulu.

```
curl -d "query=MATCH ()-[p:route]->() RETURN p LIMIT 10" -k https://localhost:8182/
openCypher -d "explain=details"
                                        ~
Query:
MATCH ()-[p:route]->() RETURN p LIMIT 10

##########################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
```

```
#################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
#################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 10        # 0.00  # 5.00      #
#################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?p]          # id2value_opencypher #
 10       # 10        # 1.00  # 1.00      #
#################################################################################################


subQuery1
#################################################################################################
# ID # Out #1 # Out #2 # Name            # Arguments
              # Mode # Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # DFEPipelineScan # pattern=Edge((?anon_node7)-[?p:?p_type1]->(?
anon_node21)) # -    # 0        # 1000      # 0.00  # 0.66      #
#    #        #        #                 # inlineFilters=[(?p_type1 IN [<el://route>])]
              #      #          #           #       #           #
#    #        #        #                 # patternEstimate=26219
              #      #          #           #       #           #
#################################################################################################
# 1  # 2      # -      # DFEProject      # columns=[?p]
              # -    # 1000     # 1000      # 1.00  # 0.14      #
#################################################################################################
# 2  # -      # -      # DFEDrain        # limit=10
              # -    # 1000     # 0         # 0.00  # 0.11      #
#################################################################################################
```

Contoh **`explain`** output untuk fungsi ekspresi nilai

Fungsinya adalah:

```
MATCH (a) RETURN DISTINCT labels(a)
```

Pada `explain` output di bawah ini, `DFEPipelineScan` (ID 0) memindai semua label node. Ini sesuai dengan`MATCH (a)`.

`DFEChunkLocalSubquery`(ID 1) mengumpulkan label `?a` untuk masing-masing`?a`. Ini sesuai dengan`labels(a)`. Anda dapat melihatnya melalui `DFEApply` dan`DFEReduce`.

`BindRelation`(ID 2) digunakan untuk mengganti nama kolom generik `?__gen_labelsOfa2` menjadi. `?labels(a)`

`DFEDistinctRelation`(ID 4) hanya mengambil label yang berbeda (multiple:airport node akan memberikan label duplikat (a): ["airport"]). Ini sesuai dengan`DISTINCT labels(a)`.

```
curl -d "query=MATCH (a) RETURN DISTINCT labels(a)" -k https://localhost:8182/
openCypher -d "explain=details"
                                          ~
Query:
MATCH (a) RETURN DISTINCT labels(a)

#################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
#################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 5         # 0.00  # 81.00     #
#################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?labels(a)]  # id2value_opencypher #
 5        # 5         # 1.00  # 1.00      #
#################################################################################################


subQuery1
#################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?a) with property 'ALL'
 and label '?a_label1'                                             # -    # 0
    # 3750      # 0.00  # 26.77     #
#    #        #        #                       # patternEstimate=3506
                                                                 #      #
  #           #       #           #
#################################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#8b314f55-2cc7-456a-a48a-c76a0465cfab/graph_1 # -    #
 3750     # 3750      # 1.00  # 0.04      #
```

```
##################################################################################################
# 2  # 3      # -      # DFEBindRelation       # inputVars=[?a, ?__gen_labelsOfa2, ?
__gen_labelsOfa2]                                                    # -    # 3750
     # 3750      # 1.00  # 0.08      #
#    #        #        #                       # outputVars=[?a, ?__gen_labelsOfa2, ?
labels(a)]                                                           #      #
    #           #       #           #
##################################################################################################
# 3  # 4      # -      # DFEProject            # columns=[?labels(a)]
                                                                     # -    # 3750
  # 3750      # 1.00  # 0.05      #
##################################################################################################
# 4  # 5      # -      # DFEDistinctRelation   # -
                                                                     # -    # 3750
  # 5         # 0.00  # 2.78      #
##################################################################################################
# 5  # -      # -      # DFEDrain              # -
                                                                     # -    # 5
  # 0         # 0.00  # 0.03      #
##################################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#8b314f55-2cc7-456a-
a48a-c76a0465cfab/graph_1
##################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                    # Mode     # Units In # Units Out # Ratio # Time (ms) #
##################################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?a]
                    # -        # 0        # 3750      # 0.00  # 0.02      #
##################################################################################################
# 1  # 2      # 3      # DFETee               # -
                    # -        # 3750     # 7500      # 2.00  # 0.02      #
##################################################################################################
# 2  # 4      # -      # DFEProject           # columns=[?a]
                    # -        # 3750     # 3750      # 1.00  # 0.04      #
##################################################################################################
# 3  # 17     # -      # DFEOptionalJoin      # -
                    # -        # 7500     # 3750      # 0.50  # 0.44      #
##################################################################################################
# 4  # 5      # -      # DFEDistinctRelation  # -
                    # -        # 3750     # 3750      # 1.00  # 2.23      #
##################################################################################################
```

```
# 5  # 6      # -      # DFEDistinctColumn    # column=?a
                     # -        # 3750     # 3750      # 1.00  # 1.50      #
#    #        #        #                      # ordered=false
                     #          #          #           #       #           #
##########################################################################################
# 6  # 7      # -      # DFEPipelineJoin      # pattern=Node(?a) with property 'ALL'
 and label '?a_label3' # -        # 3750     # 3750      # 1.00  # 10.58     #
#    #        #        #                      # patternEstimate=3506
                     #          #          #           #       #           #
##########################################################################################
# 7  # 8      # 9      # DFETee               # -
                     # -        # 3750     # 7500      # 2.00  # 0.02      #
##########################################################################################
# 8  # 10     # -      # DFEBindRelation      # inputVars=[?a_label3]
                     # -        # 3750     # 3750      # 1.00  # 0.04      #
#    #        #        #                      # outputVars=[?100]
                     #          #          #           #       #           #
##########################################################################################
# 9  # 11     # -      # DFEBindRelation      # inputVars=[?a, ?a_label3, ?100]
                     # -        # 7500     # 3750      # 0.50  # 0.07      #
#    #        #        #                      # outputVars=[?a, ?a_label3, ?100]
                     #          #          #           #       #           #
##########################################################################################
# 10 # 9      # -      # DFETermResolution    # column=?100
                     # id2value # 3750     # 3750      # 1.00  # 7.60      #
##########################################################################################
# 11 # 12     # -      # DFEBindRelation      # inputVars=[?a, ?a_label3, ?100]
                     # -        # 3750     # 3750      # 1.00  # 0.06      #
#    #        #        #                      # outputVars=[?a, ?100, ?a_label3]
                     #          #          #           #       #           #
##########################################################################################
# 12 # 13     # -      # DFEApply             # functor=nodeLabel(?a_label3)
                     # -        # 3750     # 3750      # 1.00  # 0.55      #
##########################################################################################
# 13 # 14     # -      # DFEProject           # columns=[?a, ?a_label3_alias4]
                     # -        # 3750     # 3750      # 1.00  # 0.05      #
##########################################################################################
# 14 # 15     # -      # DFEMergeChunks       # -
                     # -        # 3750     # 3750      # 1.00  # 0.02      #
##########################################################################################
# 15 # 16     # -      # DFEReduce            # functor=collect(?a_label3_alias4)
                     # -        # 3750     # 3750      # 1.00  # 6.37      #
#    #        #        #                      # segmentationKey=[?a]
                     #          #          #           #       #           #
```

```
#########################################################################################
# 16 # 3      # -      # DFEMergeChunks       # -
                    # -        # 3750     # 3750      # 1.00  # 0.03      #
#########################################################################################
# 17 # -      # -      # DFEDrain             # -
                    # -        # 3750     # 0         # 0.00  # 0.02      #
#########################################################################################
```

Contoh **`explain`** output untuk fungsi ekspresi nilai matematika

Dalam contoh ini, `RETURN abs(-10)` melakukan evaluasi sederhana, mengambil nilai absolut dari konstanta, `-10`.

`DFEChunkLocalSubQuery`(ID 1) melakukan injeksi solusi untuk nilai statis `-10`, yang disimpan dalam variabel, `?100`.

`DFEApply`(ID 2) adalah operator yang menjalankan fungsi nilai absolut `abs()` pada nilai statis yang disimpan dalam `?100` variabel.

Berikut adalah query dan `explain` output yang dihasilkan:

```
curl -d "query=RETURN abs(-10)" -k https://localhost:8182/openCypher  -d
 "explain=details"
                                                    ~
Query:
RETURN abs(-10)

#########################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments             # Mode
  # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]        # -
  # 0        # 1        # 0.00  # 0         #
#########################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1    # -
  # 0        # 1        # 0.00  # 4.00      #
#########################################################################################
# 2  # -      # -      # TermResolution    # vars=[?_internalVar1] #
 id2value_opencypher # 1        # 1         # 1.00  # 1.00      #
#########################################################################################


subQuery1
```

```
################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                    # Mode # Units
 In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFESolutionInjection  # outSchema=[]
                                                                  # -    # 0
  # 1         # 0.00  # 0.01      #
################################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#c4cc6148-cce3-4561-93c0-deb91f257356/graph_1 # -    #
 1        # 1         # 1.00  # 0.03      #
################################################################################################
# 2  # 3      # -      # DFEApply             # functor=abs(?100)
                                                                  # -    # 1
  # 1         # 1.00  # 0.26      #
################################################################################################
# 3  # 4      # -      # DFEBindRelation       # inputVars=[?_internalVar2, ?
_internalVar2]                                                                  # -
 # 1        # 1         # 1.00  # 0.04      #
#    #        #        #                      # outputVars=[?_internalVar2, ?
_internalVar1]                                                                #
 #          #           #       #           #
################################################################################################
# 4  # 5      # -      # DFEProject            # columns=[?_internalVar1]
                                                                  # -    # 1
  # 1         # 1.00  # 0.06      #
################################################################################################
# 5  # -      # -      # DFEDrain              # -
                                                                  # -    # 1
  # 0         # 0.00  # 0.05      #
################################################################################################

subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#c4cc6148-
cce3-4561-93c0-deb91f257356/graph_1
################################################################################################
# ID # Out #1 # Out #2 # Name                 # Arguments                          #
 Mode # Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFESolutionInjection # solutions=[?100 -> [-10^^<LONG>]]   # -
    # 0        # 1         # 0.00  # 0.01      #
#    #        #        #                      # outSchema=[?100]                   #
    #          #           #       #           #
################################################################################################
```

```
# 1  # 3      # -      # DFERelationalJoin    # joinVars=[]                          # -
    # 2        # 1         # 0.50  # 0.18      #
#################################################################################################
# 2  # 1      # -      # DFESolutionInjection # outSchema=[]                        # -
    # 0        # 1         # 0.00  # 0.01      #
#################################################################################################
# 3  # -      # -      # DFEDrain             # -                                   # -
    # 1        # 0         # 0.00  # 0.02      #
#################################################################################################
```

Contoh **`explain`** output untuk kueri jalur panjang variabel (VLP)

Ini adalah contoh rencana kueri yang lebih kompleks untuk menangani kueri jalur panjang variabel. Contoh ini hanya menunjukkan bagian dari `explain` output, untuk kejelasan.

Dalam`subQuery1`, `DFEPipelineScan` (ID 0) dan `DFEChunkLocalSubQuery` (ID 1), yang menyuntikkan `...graph_1` subquery, bertanggung jawab untuk memindai node dengan kode. `YPO`

Dalam`subQuery1`, `DFEChunkLocalSubQuery` (ID 2), yang menyuntikkan `...graph_2` subquery, bertanggung jawab untuk memindai node dengan kode. `LAX`

Dalam`subQuery1`, `DFEChunkLocalSubQuery` (ID 3) menyuntikkan `...graph3` subquery, yang berisi `DFELoopSubQuery` (ID 17), yang pada gilirannya menyuntikkan subquery. `...graph5` Operasi ini bertanggung jawab untuk menyelesaikan pola `-[*2]->` panjang variabel dalam string kueri antara dua node.

```
curl -d "query=MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p" -k https://
localhost:8182/openCypher -d "explain=details"


                                                ~
Query:
MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p

#################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
#################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 0         # 0.00  # 84.00     #
#################################################################################################
```

```
# 2  # -      # -      # TermResolution    # vars=[?p]          # id2value_opencypher #
 0        # 0         # 0.00  # 0         #
#########################################################################################


subQuery1
#########################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                    # Mode # Units
 In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?a) with property 'code'
 as ?a_code7 and label 'ALL'                                         # -    # 0
   # 1         # 0.00  # 0.68      #
#    #        #        #                      # inlineFilters=[(?a_code7 IN
 ["YPO"^^xsd:string])]                                                  #
  #          #           #       #           #
#    #        #        #                        # patternEstimate=1
                                                                  #      #
  #           #       #           #
#########################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_1 # -    #
 1        # 1         # 1.00  # 0.03      #
#########################################################################################
# 2  # 3      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_2 # -    #
 1        # 1         # 1.00  # 0.02      #
#########################################################################################
# 3  # 4      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_3 # -    #
 1        # 0         # 0.00  # 0.04      #
#########################################################################################
# 4  # 5      # -      # DFEBindRelation       # inputVars=[?__gen_path6, ?
anon_rel26, ?b_code8, ?b, ?a_code7, ?a, ?__gen_path6]                    # -
   # 0        # 0         # 0.00  # 0.10      #
#    #        #        #                      # outputVars=[?__gen_path6, ?
anon_rel26, ?b_code8, ?b, ?a_code7, ?a, ?p]                              #
  #          #           #       #           #
#########################################################################################
# 5  # 6      # -      # DFEProject            # columns=[?p]
                                                                  # -    # 0
  # 0         # 0.00  # 0.05      #
#########################################################################################
```

```
# 6  # -      # -      # DFEDrain              # -
                                                                    # -    # 0
  # 0         # 0.00  # 0.02      #
#########################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_1
#########################################################################################
# ID # Out #1 # Out #2 # Name                 # Arguments
                    # Mode # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?a, ?a_code7]
                    # -    # 0        # 1         # 0.00  # 0.01      #
#########################################################################################
# 1  # 2      # 3      # DFETee               # -
                    # -    # 1        # 2         # 2.00  # 0.01      #
#########################################################################################
# 2  # 4      # -      # DFEDistinctColumn    # column=?a
                    # -    # 1        # 1         # 1.00  # 0.25      #
#    #        #        #                      # ordered=false
                    #      #          #           #       #           #
#########################################################################################
# 3  # 5      # -      # DFEHashIndexBuild    # vars=[?a]
                    # -    # 1        # 1         # 1.00  # 0.05      #
#########################################################################################
# 4  # 5      # -      # DFEPipelineJoin      # pattern=Node(?a) with property 'ALL'
 and label '?a_label1' # -    # 1        # 1         # 1.00  # 0.47      #
#    #        #        #                      # patternEstimate=3506
                    #      #          #           #       #           #
#########################################################################################
# 5  # 6      # 7      # DFESync              # -
                    # -    # 2        # 2         # 1.00  # 0.04      #
#########################################################################################
# 6  # 8      # -      # DFEForwardValue      # -
                    # -    # 1        # 1         # 1.00  # 0.01      #
#########################################################################################
# 7  # 8      # -      # DFEForwardValue      # -
                    # -    # 1        # 1         # 1.00  # 0.01      #
#########################################################################################
# 8  # 9      # -      # DFEHashIndexJoin     # -
                    # -    # 2        # 1         # 0.50  # 0.26      #
#########################################################################################
```

```
# 9  # -      # -      # DFEDrain             # -
                    # -    # 1        # 0         # 0.00  # 0.02      #
##############################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_2
##############################################################################################
# ID # Out #1 # Out #2 # Name                 # Arguments
                           # Mode # Units In # Units Out # Ratio # Time (ms) #
##############################################################################################
# 0  # 1      # -      # DFEPipelineScan      # pattern=Node(?b) with property 'code'
 as ?b_code8 and label 'ALL' # -    # 0        # 1         # 0.00  # 0.38      #
#    #        #        #                      # inlineFilters=[(?b_code8 IN
 ["LAX"^^xsd:string])]                #      #          #           #       #
  #
#    #        #        #                      # patternEstimate=1
                           #      #          #           #       #           #
##############################################################################################
# 1  # 2      # -      # DFEMergeChunks       # -
                           # -    # 1        # 1         # 1.00  # 0.02      #
##############################################################################################
# 2  # 4      # -      # DFERelationalJoin    # joinVars=[]
                           # -    # 2        # 1         # 0.50  # 0.19      #
##############################################################################################
# 3  # 2      # -      # DFESolutionInjection # outSchema=[?a, ?a_code7]
                           # -    # 0        # 1         # 0.00  # 0         #
##############################################################################################
# 4  # -      # -      # DFEDrain             # -
                           # -    # 1        # 0         # 0.00  # 0.01      #
##############################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_3
##############################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode     #
 Units In # Units Out # Ratio # Time (ms) #
##############################################################################################
...
# 17 # 18     # -      # DFELoopSubQuery       # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_5 # -
  # 1        # 2         # 2.00  # 0.31      #
```

```
...
```

Transaksi di Neptunus OpenCypher

Implementasi OpenCypher di Amazon Neptunus menggunakan semantik transaksi yang ditentukan oleh Neptunus Namun, tingkat isolasi yang disediakan oleh driver Bolt memiliki beberapa implikasi spesifik untuk semantik transaksi Bolt, seperti yang dijelaskan pada bagian di bawah ini.

Kueri transaksi Bolt hanya-baca

Ada berbagai cara agar kueri hanya-baca dapat diproses, dengan model transaksi dan tingkat isolasi yang berbeda, sebagai berikut:

Kueri transaksi hanya-baca implisit

Berikut adalah contoh transaksi implisit read-only:

```
public void executeReadImplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
          Config.builder().withEncryption()
                          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                          .build());

  // create the session config
  SessionConfig sessionConfig = SessionConfig.builder()
                                             .withFetchSize(1000)
                                             .withDefaultAccessMode(AccessMode.READ)
                                             .build();

  // run the query as access mode read
  driver.session(sessionConfig).readTransaction(new TransactionWork<String>()
    {
      final StringBuilder resultCollector = new StringBuilder();

      @Override
```

```
      public String execute(final Transaction tx)
      {
        // execute the query
        Result queryResult = tx.run(READ_QUERY);

        // Read the result
        for (Record record : queryResult.list())
        {
          for (String key : record.keys())
          {
            resultCollector.append(key)
                           .append(":")
                           .append(record.get(key).asNode().toString());
          }
        }
        return resultCollector.toString();
      }

    }
  );

  // close the driver.
  driver.close();
}
```

Karena replika baca hanya menerima kueri hanya-baca, semua kueri terhadap replika baca dijalankan sebagai transaksi baca-implisit terlepas dari mode akses yang ditetapkan dalam konfigurasi sesi. Neptunus mengevaluasi transaksi baca-implisit sebagai kueri hanya-baca di bawah semantik isolasi. `SNAPSHOT`

Jika terjadi kegagalan, transaksi baca-implisit dicoba ulang secara default.

Kueri transaksi hanya-baca otomatis

Berikut adalah contoh transaksi autocommit read-only:

```
public void executeAutoCommitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";
```

```
  // Create the session config.
  final SessionConfig sessionConfig = SessionConfig
    .builder()
    .withFetchSize(1000)
    .withDefaultAccessMode(AccessMode.READ)
    .build();

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
    Config.builder()
          .withEncryption()
          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
          .build());

  // result collector
  final StringBuilder resultCollector = new StringBuilder();

  // create a session
  final Session session = driver.session(sessionConfig);

  // run the query
  final Result queryResult = session.run(READ_QUERY);
  for (final Record record : queryResult.list())
  {
    for (String key : record.keys())
    {
      resultCollector.append(key)
                     .append(":")
                     .append(record.get(key).asNode().toString());
    }
  }

  // close the session
  session.close();

  // close the driver
  driver.close();
}
```

Jika mode akses diatur ke READ dalam konfigurasi sesi, Neptunus mengevaluasi kueri transaksi autocommit sebagai kueri hanya-baca di bawah semantik isolasi. SNAPSHOT Perhatikan bahwa replika baca hanya menerima kueri hanya-baca.

Jika Anda tidak meneruskan konfigurasi sesi, kueri komit otomatis diproses secara default dengan isolasi kueri mutasi, jadi penting untuk meneruskan konfigurasi sesi yang secara eksplisit menyetel mode akses ke. READ

Jika terjadi kegagalan, kueri komit otomatis hanya-baca tidak dicoba ulang.

Kueri transaksi hanya-baca eksplisit

Berikut adalah contoh transaksi hanya-baca eksplisit:

```
public void executeReadExplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";

  // Create the session config.
  final SessionConfig sessionConfig = SessionConfig
    .builder()
    .withFetchSize(1000)
    .withDefaultAccessMode(AccessMode.READ)
    .build();

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
    Config.builder()
          .withEncryption()
          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
          .build());

  // result collector
  final StringBuilder resultCollector = new StringBuilder();

  // create a session
  final Session session = driver.session(sessionConfig);

  // begin transaction
  final Transaction tx = session.beginTransaction();

  // run the query on transaction
  final List<Record> list = tx.run(READ_QUERY).list();
```

```
  // read the result
  for (final Record record : list)
  {
    for (String key : record.keys())
    {
      resultCollector
        .append(key)
        .append(":")
        .append(record.get(key).asNode().toString());
    }
  }

  // commit the transaction and for rollback we can use beginTransaction.rollback();
  tx.commit();

  // close the driver
  driver.close();
}
```

Jika mode akses diatur ke READ dalam konfigurasi sesi, Neptunus mengevaluasi transaksi hanya-baca eksplisit sebagai kueri hanya-baca di bawah semantik isolasi. SNAPSHOT Perhatikan bahwa replika baca hanya menerima kueri hanya-baca.

Jika Anda tidak meneruskan konfigurasi sesi, transaksi hanya-baca eksplisit diproses secara default dengan isolasi kueri mutasi, jadi penting untuk meneruskan konfigurasi sesi yang secara eksplisit menyetel mode akses ke. READ

Jika terjadi kegagalan, kueri eksplisit hanya-baca akan dicoba ulang secara default.

Kueri transaksi Mutasi Bolt

Seperti halnya kueri hanya-baca, ada berbagai cara agar kueri mutasi dapat diproses, dengan model transaksi dan tingkat isolasi yang berbeda, sebagai berikut:

Kueri transaksi mutasi implisit

Berikut adalah contoh transaksi mutasi implisit:

```
public void executeWriteImplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";
```

```
// create node with label as label and properties.
final String WRITE_QUERY = "CREATE (n:label {name : 'foo'})";

// Read the vertex created with label as label.
final String READ_QUERY = "MATCH (n:label) RETURN n";

// create the driver
final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
  Config.builder()
        .withEncryption()
        .withTrustStrategy(TrustStrategy.trustSystemCertificates())
        .build());

// create the session config
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();

final StringBuilder resultCollector = new StringBuilder();

// run the query as access mode write
driver.session(sessionConfig).writeTransaction(new TransactionWork<String>()
{
  @Override
  public String execute(final Transaction tx)
  {
    // execute the write query and consume the result.
    tx.run(WRITE_QUERY).consume();

    // read the vertex written in the same transaction
    final List<Record> list = tx.run(READ_QUERY).list();

    // read the result
    for (final Record record : list)
    {
      for (String key : record.keys())
      {
        resultCollector
          .append(key)
          .append(":")
          .append(record.get(key).asNode().toString());
      }
```

```
      }
      return resultCollector.toString();
    }
  }); // at the end, the transaction is automatically committed.

  // close the driver.
  driver.close();
}
```

Pembacaan yang dibuat sebagai bagian dari kueri mutasi dijalankan secara `READ_COMMITTED` terpisah dengan jaminan biasa untuk transaksi mutasi Neptunus.

Apakah Anda secara khusus lulus dalam konfigurasi sesi, transaksi selalu diperlakukan sebagai transaksi tulis.

Untuk konflik, lihatResolusi Konflik Menggunakan Lock-Wait Timeout.

Kueri transaksi mutasi komit otomatis

Kueri autocommit mutasi mewarisi perilaku yang sama dengan transaksi implisit mutasi.

Jika Anda tidak lulus dalam konfigurasi sesi, transaksi diperlakukan sebagai transaksi tulis secara default.

Jika terjadi kegagalan, kueri komit otomatis mutasi tidak dicoba ulang secara otomatis.

Kueri transaksi mutasi eksplisit

Berikut adalah contoh transaksi mutasi eksplisit:

```
public void executeWriteExplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // create node with label as label and properties.
  final String WRITE_QUERY = "CREATE (n:label {name : 'foo'})";

  // Read the vertex created with label as label.
  final String READ_QUERY = "MATCH (n:label) RETURN n";

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
    Config.builder()
```

```
        .withEncryption()
        .withTrustStrategy(TrustStrategy.trustSystemCertificates())
        .build());

// create the session config
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();

final StringBuilder resultCollector = new StringBuilder();

final Session session = driver.session(sessionConfig);

// run the query as access mode write
final Transaction tx = driver.session(sessionConfig).beginTransaction();

// execute the write query and consume the result.
tx.run(WRITE_QUERY).consume();

// read the result from the previous write query in a same transaction.
final List<Record> list = tx.run(READ_QUERY).list();

// read the result
for (final Record record : list)
{
  for (String key : record.keys())
  {
    resultCollector
      .append(key)
      .append(":")
      .append(record.get(key).asNode().toString());
  }
}

// commit the transaction and for rollback we can use tx.rollback();
tx.commit();

// close the session
session.close();

// close the driver.
driver.close();
```

```
}
```

Kueri mutasi eksplisit mewarisi perilaku yang sama dengan transaksi mutasi implisit.

Jika Anda tidak lulus dalam konfigurasi sesi, transaksi diperlakukan sebagai transaksi tulis secara default.

Untuk konflik, lihatResolusi Konflik Menggunakan Lock-Wait Timeout.

Petunjuk kueri OpenCypher

> **Important**
>
> Petunjuk kueri OpenCypher hanya tersedia dari rilis mesin 1.3.2.0 dan yang lebih baru.

Di Amazon Neptunus, Anda dapat menggunakan `USING` klausa untuk menentukan petunjuk kueri untuk kueri OpenCypher. Petunjuk ini memungkinkan Anda untuk mengontrol strategi pengoptimalan dan evaluasi.

Sintaks untuk petunjuk kueri adalah:

```
USING {scope}:{hint} {value}
```

1. `{scope}`mendefinisikan ruang lingkup di mana petunjuk berlaku untuk: `Query` atau. `Clause`

   Nilai lingkup `Query` berarti bahwa petunjuk kueri berlaku untuk seluruh kueri (tingkat kueri).

   Nilai cakupan `Clause` berarti bahwa petunjuk kueri berlaku untuk klausa yang didahului petunjuk (tingkat klausa).
2. `{hint}`adalah nama petunjuk kueri yang diterapkan.
3. `{value}`adalah argumen untuk`{hint}`.

Nilainya bisa tidak peka huruf besar/kecil.

Misalnya, untuk mengaktifkan cache rencana kueri untuk kueri:

```
Using QUERY:PLANCACHE "enabled"
MATCH (a:Person {firstName: "Erin", lastName: $lastName})
```

```
RETURN a
```

> **Note**
>
> Saat ini, petunjuk kueri lingkup Kueri PLANCACHE, TIMEOUTMILLISECONDS, dan Jenis didukung. assumeConsistentData Petunjuk kueri yang didukung tercantum di bawah ini.

Topik

- Petunjuk cache rencana kueri OpenCypher
- AssumeConsistentDataTypes petunjuk
- Petunjuk batas waktu kueri OpenCypher

Petunjuk cache rencana kueri OpenCypher

Perilaku cache rencana kueri dapat diganti berdasarkan per-kueri (diparameterisasi atau tidak) dengan petunjuk kueri tingkat kueri. `QUERY:PLANCACHE` Itu perlu digunakan dengan `USING` klausa. Petunjuk kueri menerima `enabled` atau `disabled` sebagai nilai. Untuk informasi selengkapnya tentang cache paket kueri, lihatCache paket kueri di Amazon Neptunus.

```
# Forcing plan to be cached or reused
% curl -k https://<endpoint>:<port>/opencypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1"

% curl -k https://<endpoint>:<port>/opencypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" RETURN \$arg" \
  -d "parameters={\"arg\": 123}"

# Forcing plan to be neither cached nor reused
% curl -k https://<endpoint>:<port>/opencypher \
  -d "query=Using QUERY:PLANCACHE \"disabled\" MATCH(n) RETURN n LIMIT 1"
```

AssumeConsistentDataTypes petunjuk

OpenCypher mengikuti paradigma di mana kecocokan tipe data numerik (misalnya, int, byte, pendek, panjang, dll.) dilakukan di bawah semantik promosi tipe. Misalnya, ketika mencari semua properti dengan nilai input 10 dengan tipe pendek, di bawah semantik promosi tipe, itu juga akan cocok dengan properti yang memiliki 10 sebagai nilai panjang. Dalam beberapa kasus, pengecoran tipe

dapat menyebabkan overhead dan menghasilkan rencana kueri yang kurang efisien daripada jika tidak ada pengecoran tipe yang dilakukan. Khususnya dalam kasus di mana tipe data digunakan secara konsisten dalam data (misalnya, jika semua usia orang disimpan sebagai nilai panjang) melakukan promosi tipe menyebabkan overhead tanpa memengaruhi hasil kueri.

Untuk memungkinkan pengoptimalan kasus ketika diketahui bahwa nilai data properti numerik yang disimpan dalam database adalah tipe yang konsisten, petunjuk kueri yang disebut `assumeConsistentDataTypes` (dengan nilai`true/false`, dengan default`false`) dapat digunakan. Ketika petunjuk kueri ini diberikan dengan nilai`true`, mesin mengasumsikan satu-satunya nilai properti selalu panjang atau ganda dan akan melewati semantik promosi jenis. Nilai numerik yang ditentukan dalam kueri dianggap sebagai nilai panjang (untuk nilai non-floating point) dan ganda (untuk nilai floating point).

Jika data secara konsisten menggunakan tipe data tunggal (misalnya semua usia disimpan sebagai`long`), maka menggunakan `assumeConsistentDataTypes` petunjuk dapat mengoptimalkan kueri dengan melewatkan pemeriksaan kesetaraan yang tidak perlu untuk jenis numerik yang berbeda. Namun, jika data memiliki tipe data yang tidak konsisten untuk properti yang sama, maka menggunakan petunjuk dapat menyebabkan beberapa hasil terlewatkan, karena kueri hanya akan cocok dengan tipe data tunggal yang diasumsikan oleh petunjuk.

```
# Database loaded with following openCypher CSV's

# File 1
:ID,age:Int
n1,20
n2,25

# File 2
:ID,age:Long
n3,25


# Example (no hint)
MATCH (n:Person)
WHERE n.age >= 25
RETURN n

# Result
n2
n3
```

```
Returns all person whose age is >= 25 and the values >= 25 can be with any of these
 datatypes
i.e. byte, short, int, long, double or float

-------------------------------------------------------------------------------

# Example (with hint present)
USING QUERY:assumeConsistentDataTypes "true"
MATCH (n:Person)
WHERE n.age >= 25
RETURN n

# Result
n3

Returns only "n3" and not "n2". The reason is that even though the numerical value
matches (25), the datatype is "int" and is considered a non-match.
```

Perbedaannya juga dapat divalidasi melalui penjelasan.

Tanpa penjelasan:

```
# Query
MATCH (n)
WHERE n.age = 20
RETURN n

# Explain Snippet
###############################################################################################
# ID # Out #1 # Out #2 # Name                   # Arguments

         # Mode # Units In # Units Out # Ratio # Time (ms) #
###############################################################################################
# 0  # 1      # -      # DFEPipelineScan (DFX)  # pattern=Node(?n) with property 'age'
 as ?n_age2 and label 'ALL'
          # -    # 0        # 1         # 0.00  # 0.10      #
#    #        #        #                       # inlineFilters=[(?n_age2 IN
 ["20"^^xsd:byte, "20"^^xsd:int, "20"^^xsd:long, "20"^^xsd:short, "20.0"^^xsd:double,
 "20.0"^^xsd:float])] #      #          #           #       #           #
#    #        #        #                       # patternEstimate=1

         #      #          #           #       #           #
###############################################################################################
```

```
# The inFilters field contains all numeric types
```

Dengan petunjuk:

```
# Query
MATCH (n)
WHERE n.age = 20
RETURN n

# Explain Snippet
################################################################################################
# ID # Out #1 # Out #2 # Name                    # Arguments
                          # Mode # Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFEPipelineScan (DFX)  # pattern=Node(?n) with property 'age'
 as ?n_age2 and label 'ALL' # -    # 0        # 1         # 0.00  # 0.07      #
#    #        #        #                        # inlineFilters=[(?n_age2 IN
 ["20"^^xsd:long])]                  #      #          #           #       #
 #
#    #        #        #                        # patternEstimate=1
                          #      #          #           #       #           #
################################################################################################

# The inFilters field only contains long datatype
```

Petunjuk batas waktu kueri OpenCypher

Perilaku batas waktu kueri dapat dikonfigurasi berdasarkan per-kueri dengan petunjuk kueri tingkat kueri. `QUERY:TIMEOUTMILLISECONDS` Itu harus digunakan dengan `USING` klausa. Petunjuk kueri menerima panjang non-negatif sebagai nilai.

```
# Using query-level timeout hint

% curl https://<endpoint>:<port>/opencypher \
  -d "query=USING QUERY:TIMEOUTMILLISECONDS 100 MATCH(n) RETURN n LIMIT 1"
```

Perilaku batas waktu kueri akan mempertimbangkan batas waktu minimum tingkat cluster dan batas waktu tingkat kueri. Silakan lihat contoh di bawah ini untuk memahami perilaku batas waktu kueri. Untuk informasi selengkapnya tentang batas waktu kueri tingkat cluster, lihat neptune_query_timeout.

```
# Suppose `neptune_query_timeout` is 10000 ms and query-level timeout is set to 100 ms
# It will consider 100 ms as the final timeout
```

```
% curl https://<endpoint>:<port>/opencypher \
  -d "query=USING QUERY:TIMEOUTMILLISECONDS 100 MATCH(n) RETURN n LIMIT 1"

# Suppose `neptune_query_timeout` is 100 ms and query-level timeout is set to 10000 ms
# It will still consider 100 ms as the final timeout

% curl https://<endpoint>:<port>/opencypher \
  -d "query=USING QUERY:TIMEOUTMILLISECONDS 10000 MATCH(n) RETURN n LIMIT 1"
```

Pembatasan Neptunus OpenCypher

Rilis Amazon Neptunus dari OpenCypher masih tidak mendukung semua yang ditentukan dalam Referensi Bahasa Kueri Cypher, Versi 9, seperti yang dirinci dalam. Kepatuhan spesifikasi OpenCypher Rilis masa depan diharapkan dapat mengatasi banyak keterbatasan tersebut.

Pengecualian Neptunus OpenCypher

Saat bekerja dengan OpenCypher di Amazon Neptunus, berbagai pengecualian dapat terjadi. Di bawah ini adalah pengecualian umum yang mungkin Anda terima, baik dari titik akhir HTTPS atau dari driver Bolt (semua pengecualian dari driver Bolt dilaporkan sebagai Pengecualian Status Server):

Kode HTTP	Pesan kesalahan	Dapat diambil?	Tindakan Perbaikan	
400	(kesalahan sintaks, disebarkan langsung dari parser OpenCypher)	Tidak	Perbaiki sintaks kueri, lalu coba lagi.	
500	`Operation terminated (out of memory)`	Ya	Ulangi kueri untuk menambahkan kriteria penyaringan tambahan untuk mengurangi	

Kode HTTP	Pesan kesalahan	Dapat diambil?	Tindakan Perbaikan	
			memori yang diperlukan	
500	Operasi dihentikan (batas waktu terlampaui)	Ya	Tingkatkan batas waktu kueri di grup parameter cluster DB, atau coba lagi permintaannya.	
500	Operasi dihentikan (dibatalkan oleh pengguna)	Ya	Coba lagi permintaannya.	
500	Reset basis data sedang berlangsung. Silakan coba lagi kueri setelah klaster tersedia.	Ya	Coba lagi ketika reset selesai.	
500	Operasi gagal karena operasi bersamaan yang bertentangan (silakan coba lagi). Transaksi saat ini bergulir kembali.	Ya	Coba lagi menggunakan strategi backoff dan coba lagi eksponensial.	

Kode HTTP	Pesan kesalahan	Dapat diambil?	Tindakan Perbaikan	
400	*(operation name)* operation/feature Pengecualian tidak didukung	Tidak	Operasi yang ditentukan tidak didukung.	
400	Pembaruan OpenCypher dicoba pada replika hanya-baca	Tidak	Ubah titik akhir target ke titik akhir penulis.	
400	Malformed QueryException (Neptunus tidak menunjukkan status parser internal)	Tidak	Sintaks kueri yang benar dan coba lagi.	
400	Tidak dapat menghapus node, karena masih memiliki hubungan. Untuk menghapus node ini, Anda harus terlebih dahulu menghapus hubungannya.	Tidak	Alih-alih menggunakan `MATCH (n) DELETE n` penggunaan `MATCH(n) DETACH DELETE(n)`	

Kode HTTP	Pesan kesalahan	Dapat diambil?	Tindakan Perbaikan	
400	Operasi tidak valid: mencoba menghapus label terakhir dari sebuah node. Sebuah node harus memiliki setidaknya satu label.	Tidak	Neptunus mengharuskan semua node memiliki setidaknya satu label, dan jika node dibuat tanpa label eksplisit, label default ditetapkan. `vertex` Ubah logika and/or aplikasi kueri agar tidak menghapus label terakhir. Label tunggal dari sebuah node dapat diperbarui dengan menetapkan label baru dan kemudian menghapus label lama.	

Kode HTTP	Pesan kesalahan	Dapat diambil?	Tindakan Perbaikan	
500	Jumlah maksimum permintaan telah dilanggar, Configure dQueueCapacity = {} untuk ConnID = {}	Ya	Saat ini hanya 8.192 permintaan bersamaan yang dapat diproses, terlepas dari tumpukan dan protokolnya.	
500	Batas koneksi maks dilanggar.	Ya	Hanya 1000 koneksi Bolt bersamaan per instance yang diizinkan (untuk HTTP tidak ada batasan).	
400	Diharapkan [salah satu dari: Node, Relationship or Path] dan mendapat Literal	Tidak	Periksa apakah Anda meneruskan argumen yang benar, sintaks kueri yang benar, dan coba lagi.	
400	Nilai properti harus literal sederhana. Atau: Peta yang Diharapkan untuk properti Set tetapi tidak menemukannya.	Tidak	Klausa SET hanya menerima literal sederhana, bukan tipe komposit.	

Kode HTTP	Pesan kesalahan	Dapat diambil?	Tindakan Perbaikan	
400	Entitas yang ditemukan lolos untuk penghapusan tidak ditemukan	Tidak	Periksa apakah entitas yang Anda coba hapus ada di database.	
400	Pengguna tidak memiliki akses ke database.	Tidak	Periksa kebijakan tentang peran IAM yang digunakan.	
400	Tidak ada token yang dilewatkan sebagai bagian dari permintaan	Tidak	Token yang ditandatangani dengan benar harus diteruskan sebagai bagian dari permintaan kueri pada cluster yang diaktifkan IAM.	
400	Pesan kesalahan disebarkan.	Tidak	Hubungi AWS Support dengan ID Permintaan.	
500	Operasi dihentikan (kesalahan internal)	Ya	Hubungi AWS Support dengan ID Permintaan.	

Ekstensi OpenCypher di Amazon Neptunus

Amazon Neptunus mendukung referensi spesifikasi OpenCypher versi 9. Silakan lihat Kepatuhan spesifikasi OpenCypher di Amazon Neptunus di Amazon Neptunus untuk detailnya. Selain itu, Amazon Neptunus mendukung fitur yang tercantum di sini. Kecuali versi tertentu disebutkan, fitur-fiturnya tersedia di Database Neptunus dan Neptunus Analytics.

Akses data S3 waktu kueri

Tersedia dalam Database Neptunus 1.4.7.0 dan yang lebih tinggi.

Neptunus mendukung `neptune.read()` fungsi untuk membaca data CSV atau Parket dari Amazon S3 langsung dalam kueri OpenCypher. Tidak seperti pemuat massal yang mengimpor data sebelum melakukan kueri, `neptune.read()` mengakses data Amazon S3 pada waktu eksekusi kueri.

Untuk dokumentasi lengkap, lihatneptune.read ().

Fungsi khusus Neptunus **`join()`**

Tersedia dalam Database Neptunus dan Neptunus Analytics.

Neptunus mengimplementasikan fungsi `join()` yang tidak ada dalam spesifikasi OpenCypher. Ini menciptakan string literal dari daftar literal string dan pembatas string. Dibutuhkan dua argumen:

- Argumen pertama adalah daftar literal string.
- Argumen kedua adalah string pembatas, yang dapat terdiri dari nol, satu, atau lebih dari satu karakter.

Contoh:

```
join(["abc", "def", "ghi"], ", ")    // Returns "abc, def, ghi"
```

Fungsi khusus Neptunus **`removeKeyFromMap()`**

Tersedia dalam Database Neptunus dan Neptunus Analytics.

Neptunus mengimplementasikan fungsi `removeKeyFromMap()` yang tidak ada dalam spesifikasi OpenCypher. Ini menghapus kunci tertentu dari peta dan mengembalikan peta baru yang dihasilkan.

Fungsi ini membutuhkan dua argumen:

- Argumen pertama adalah peta untuk menghapus kunci.
- Argumen kedua adalah kunci untuk menghapus dari peta.

`removeKeyFromMap()`Fungsi ini sangat berguna dalam situasi di mana Anda ingin menetapkan nilai untuk node atau hubungan dengan membuka daftar peta. Contoh:

```
UNWIND [{`~id`: 'id1', name: 'john'}, {`~id`: 'id2', name: 'jim'}] as val
CREATE (n {`~id`: val.`~id`})
SET n = removeKeyFromMap(val, '~id')
```

Nilai ID kustom untuk properti node dan hubungan

Tersedia dalam Database Neptunus 1.2.0.2 dan lebih tinggi, dan Neptunus Analytics.

Mulai rilis mesin 1.2.0.2, Neptunus telah memperluas spesifikasi OpenCypher sehingga Anda sekarang dapat menentukan `id` nilai untuk node dan hubungan di,, dan klausa. `CREATE` `MERGE` `MATCH` Ini memungkinkan Anda menetapkan string yang ramah pengguna alih-alih yang dihasilkan sistem untuk mengidentifikasi node dan hubungan UUIDs .

Di Neptunus Analytics, nilai id kustom tidak tersedia untuk tepi.

> **Warning**
>
> Ekstensi ke spesifikasi OpenCypher ini tidak kompatibel ke belakang, karena sekarang `~id` dianggap sebagai nama properti yang dicadangkan. Jika Anda sudah menggunakan `~id` sebagai properti dalam data dan kueri, Anda harus memigrasikan properti yang ada ke kunci properti baru dan menghapus yang lama. Lihat Apa yang harus dilakukan jika Anda saat ini menggunakan `~id` sebagai properti.

Berikut adalah contoh yang menunjukkan cara membuat node dan relasi yang memiliki ID kustom:

```
CREATE (n {`~id`: 'fromNode', name: 'john'})
  -[:knows {`~id`: 'john-knows->jim', since: 2020}]
  ->(m {`~id`: 'toNode', name: 'jim'})
```

Jika Anda mencoba membuat ID kustom yang sudah digunakan, Neptunus akan membuat kesalahan. `DuplicateDataException`

Berikut adalah contoh penggunaan ID kustom dalam `MATCH` klausa:

```
MATCH (n {`~id`: 'id1'})
RETURN n
```

Berikut adalah contoh penggunaan kustom IDs dalam `MERGE` klausa:

```
MATCH (n {name: 'john'}), (m {name: 'jim'})
MERGE (n)-[r {`~id`: 'john->jim'}]->(m)
RETURN r
```

Apa yang harus dilakukan jika Anda saat ini menggunakan **`~id`** sebagai properti

Dengan rilis mesin 1.2.0.2, `~id` kunci dalam klausa OpenCypher sekarang diperlakukan sebagai pengganti sebagai properti. `id` Ini berarti bahwa jika Anda memiliki properti bernama`~id`, mengaksesnya menjadi tidak mungkin.

Jika Anda menggunakan `~id` properti, yang harus Anda lakukan sebelum memutakhirkan ke rilis mesin `1.2.0.2` atau di atasnya adalah terlebih dahulu memigrasikan `~id` properti yang ada ke kunci properti baru, lalu menghapus properti tersebut`~id`. Misalnya, kueri di bawah ini:

- Menciptakan properti baru bernama 'NewId' untuk semua node,
- menyalin nilai properti '~id' ke properti 'NewId',
- dan menghapus properti '~id' dari data

```
MATCH (n)
WHERE exists(n.`~id`)
SET n.newId = n.`~id`
REMOVE n.`~id`
```

Hal yang sama perlu dilakukan untuk setiap hubungan dalam data yang memiliki `~id` properti.

Anda juga harus mengubah kueri apa pun yang Anda gunakan sebagai referensi `~id` properti. Misalnya, kueri ini:

```
MATCH (n)
WHERE n.`~id` = 'some-value'
RETURN n
```

... akan berubah menjadi ini:

```
MATCH (n)
WHERE n.newId = 'some-value'
RETURN n
```

Dukungan subquery CALL di Neptunus

Tersedia dalam Database Neptunus 1.4.1.0 dan lebih tinggi, dan Neptunus Analytics.

Amazon Neptunus mendukung subkueri`CALL`. `CALL`Subquery adalah bagian dari query utama yang berjalan dalam lingkup terisolasi untuk setiap input ke `CALL` subquery.

Misalnya, grafik berisi data tentang orang, teman mereka, dan kota tempat mereka tinggal. Kita dapat mengambil dua kota terbesar tempat setiap teman seseorang tinggal dengan menggunakan `CALL` subquery:

```
MATCH (person:Person)-[:knows]->(friend)
CALL {
  WITH friend
  MATCH (friend)-[:lived_in]->(city)
  RETURN city
  ORDER BY city.population DESC
  LIMIT 2
}
RETURN person, friend, city
```

Dalam contoh ini, bagian kueri di dalam `CALL { ... }` dijalankan untuk masing-masing `friend` yang dicocokkan dengan klausa MATCH sebelumnya. Ketika kueri batin dijalankan, `LIMIT` klausa `ORDER` dan bersifat lokal di kota-kota tempat seorang teman tertentu tinggal, jadi kami memperoleh (paling banyak) dua kota per teman.

Semua klausa kueri tersedia di dalam `CALL` subkueri. Ini termasuk `CALL` subquery bersarang juga. Beberapa batasan untuk `WITH` klausa pertama dan variabel yang dipancarkan ada dan dijelaskan di bawah ini.

Lingkup variabel di dalam subquery CALL

Variabel dari klausa sebelum `CALL` subquery yang digunakan di dalamnya harus diimpor oleh klausa awal. `WITH` Tidak seperti `WITH` klausa reguler hanya dapat berisi daftar variabel tetapi tidak

mengizinkan aliasing dan tidak dapat digunakan bersama dengan`DISTINCT`,, `ORDER BY` `WHERE`, `SKIP` atau. `LIMIT`

Variabel dikembalikan dari CALL subquery

Variabel yang dipancarkan dari `CALL` subquery ditentukan dengan klausa akhir. `RETURN` Perhatikan bahwa variabel yang dipancarkan tidak dapat tumpang tindih dengan variabel sebelum subquery. `CALL`

Batasan

Sampai sekarang, pembaruan di dalam `CALL` subquery tidak didukung.

Fungsi Neptunus OpenCypher

Tersedia dalam Database Neptunus 1.4.1.0 dan lebih tinggi, dan Neptunus Analytics.

textIndexOf

`textIndexOf(text :: STRING, lookup :: STRING, from = 0 :: INTEGER?, to = -1 :: INTEGER?) :: (INTEGER?)`

Mengembalikan indeks kejadian pertama `lookup` dalam kisaran `text` mulai dari offset `from` (inklusif), melalui offset `to` (eksklusif). Jika `to` -1, rentang berlanjut hingga akhir`text`. Pengindeksan berbasis nol, dan dinyatakan dalam nilai skalar Unicode (poin kode non-pengganti).

```
RETURN textIndexOf('Amazon Neptune', 'e')
{
  "results": [{
      "textIndexOf('Amazon Neptune', 'e')": 8
    }]
}
```

collToSet

`collToSet(values :: LIST OF ANY?) :: (LIST? OF ANY?)`

Mengembalikan daftar baru yang hanya berisi elemen unik dari daftar asli. Urutan daftar asli dipertahankan (misalnya `[1, 6, 5, 1, 5]` pengembalian`[1, 6, 5]`).

```
RETURN collToSet([1, 6, 5, 1, 1, 5])
{
```

```
  "results": [{
      "collToSet([1, 6, 5, 1, 1, 5])": [1, 6, 5]
    }]
}
```

CollKurangi

`collSubtract(first :: LIST OF ANY?, second :: LIST OF ANY?) :: (LIST? OF ANY?)`

Mengembalikan daftar baru yang berisi semua elemen unik dari `first` mengecualikan elemen dari`second`.

```
RETURN collSubtract([2, 5, 1, 0], [1, 5])
{
  "results": [{
      "collSubtract([2, 5, 1, 0], [1, 5])": [0, 2]
    }]
}
```

CollIntersection

`collIntersection(first :: LIST? OF ANY?, second :: LIST? OF ANY?) :: (LIST? OF ANY?)`

Mengembalikan daftar baru yang berisi semua elemen unik dari persimpangan `first` dan`second`.

```
RETURN collIntersection([2, 5, 1, 0], [1, 5])
{
  "results": [{
      "collIntersection([2, 5, 1, 0], [1, 5])": [1, 5]
    }]
}
```

Fungsi penyortiran

Bagian berikut mendefinisikan fungsi untuk mengurutkan koleksi. Fungsi-fungsi ini mengambil (dalam beberapa kasus opsional) argumen `config` peta, atau daftar beberapa peta semacam itu, yang menentukan kunci pengurutan arah and/or pengurutan:

```
{ key: STRING, order: STRING }
```

Berikut `key` adalah properti map atau node yang nilainya akan digunakan untuk menyortir. `order`adalah "`asc`" atau "`desc`" (case insensitive) untuk menentukan jenis naik atau turun, masing-masing. Secara default, penyortiran akan dilakukan dalam urutan menaik.

CollSort

`collSort(coll :: LIST OF ANY, config :: MAP?) :: (LIST? OF ANY?)`

Mengembalikan daftar diurutkan baru yang berisi unsur-unsur dari daftar `coll` masukan.

```
RETURN collSort([5, 3, 1], {order: 'asc'})
{
  "results": [{
      "collSort([5, 3, 1])": [1, 3, 5]
    }]
}
```

collSortMaps

`collSortMaps(coll :: LIST OF MAP, config :: MAP) :: (LIST? OF ANY?)`

Mengembalikan daftar peta diurutkan berdasarkan nilai `key` properti tertentu.

```
RETURN collSortMaps([{name: 'Alice', age: 25}, {name: 'Bob', age: 35}, {name:
 'Charlie', age: 18}], {key: 'age', order: 'desc'})
{
  "results": [{
      "x": [{
          "age": 35,
          "name": "Bob"
        }, {
          "age": 25,
          "name": "Alice"
        }, {
          "age": 18,
          "name": "Charlie"
        }]
    }]
}
```

collSortMulti

```
collSortMulti(coll :: LIST OF MAP?,
```

```
configs = [] :: LIST OF MAP,
limit = -1 :: INTEGER?,
skip = 0 :: INTEGER?) :: (LIST? OF ANY?)
```

Mengembalikan daftar peta diurutkan berdasarkan nilai `key` properti tertentu, opsional menerapkan batas dan lewati.

```
RETURN collSortMulti([{name: 'Alice', age: 25}, {name: 'Bob', age: 35}, {name:
 'Charlie', age: 18}], [{key: 'age', order: 'desc'}, {key:'name'}]) as x
{
  "results": [{
      "x": [{
          "age": 35,
          "name": "Bob"
        }, {
          "age": 25,
          "name": "Alice"
        }, {
          "age": 18,
          "name": "Charlie"
        }]
    }]
}
```

collSortNodes

`collSortNodes(coll :: LIST OF NODE, config :: MAP) :: (LIST? OF NODE?)`

Mengembalikan versi diurutkan dari daftar `coll` masukan, mengurutkan elemen node dengan nilai-nilai `key` properti masing-masing.

```
create (n:person {name: 'Alice', age: 23}), (m:person {name: 'Eve', age: 21}),
 (o:person {name:'Bob', age:25})
{"results":[]}

match (n:person) with collect(n) as people return collSortNodes(people, {key: 'name',
 order: 'desc'})
{
  "results": [{
      "collSortNodes(people, 'name')": [{
          "~id": "e599240a-8c23-4337-8aa8-f603c8fb5488",
          "~entityType": "node",
          "~labels": ["person"],
```

```
          "~properties": {
            "age": 21,
            "name": "Eve"
          }
        }, {
          "~id": "8a6ef785-59e3-4a0b-a0ff-389655a9c4e6",
          "~entityType": "node",
          "~labels": ["person"],
          "~properties": {
            "age": 25,
            "name": "Bob"
          }
        }, {
          "~id": "466bc826-f47f-452c-8a27-6b7bdf7ae9b4",
          "~entityType": "node",
          "~labels": ["person"],
          "~properties": {
            "age": 23,
            "name": "Alice"
          }
        }]
    }]
}

match (n:person) with collect(n) as people return collSortNodes(people, {key: 'age'})
{
  "results": [{
      "collSortNodes(people, '^age')": [{
          "~id": "e599240a-8c23-4337-8aa8-f603c8fb5488",
          "~entityType": "node",
          "~labels": ["person"],
          "~properties": {
            "age": 21,
            "name": "Eve"
          }
        }, {
          "~id": "466bc826-f47f-452c-8a27-6b7bdf7ae9b4",
          "~entityType": "node",
          "~labels": ["person"],
          "~properties": {
            "age": 23,
            "name": "Alice"
          }
        }, {
```

```
          "~id": "8a6ef785-59e3-4a0b-a0ff-389655a9c4e6",
          "~entityType": "node",
          "~labels": ["person"],
          "~properties": {
            "age": 25,
            "name": "Bob"
          }
        }]
    }]
}
```

Fungsi temporal

Fungsi temporal tersedia dari Neptunus versi 1.4.5.0 dan yang lebih tinggi.

hari

`day(temporal :: (datetime | date)) :: (LONG)`

`day`Mengembalikan bulan dari `date` nilai `datetime` atau. Untuk`datetime`: nilai dinormalisasi ke UTC berdasarkan input sebelum mengekstraksi hari. Untuk`date`: hari diekstraksi berdasarkan zona waktu.

`datetime`Masukan tersedia di Database Neptunus dan Neptunus Analytics:

```
RETURN day(datetime('2021-06-03T01:48:14Z'))
{
  "results": [{
      "day(datetime('2021-06-03T01:48:14Z'))": 3
    }]
}
```

Di sini, `datetime` dinormalisasi ke UTC, jadi + 08:00 bergeser kembali ke 2 Juni.

```
RETURN day(datetime('2021-06-03T00:00:00+08:00'))
{
  "results": [{
      "day(datetime('2021-06-03T00:00:00+08:00'))": 2
    }]
}
```

`date`Masukan hanya tersedia di Neptunus Analytics:

```
RETURN day(date('2021-06-03Z'))
{
  "results": [{
      "day(date('2021-06-03Z'))": 3
    }]
}
```

Zona waktu `date` mempertahankan, menjaga 3 Juni.

```
RETURN day(date('2021-06-03+08:00'))
{
  "results": [{
      "day(date('2021-06-03+08:00'))": 3
    }]
}
```

bulan

`month(temporal :: (datetime | date)) :: (LONG)`

Mengembalikan bulan dari `date` nilai `datetime` atau (1-12). Untuk`datetime`: nilai dinormalisasi ke UTC berdasarkan input sebelum mengekstraksi bulan. Untuk`date`: bulan diekstraksi berdasarkan zona waktu.

`datetime`Masukan tersedia di Database Neptunus dan Neptunus Analytics:

```
RETURN month(datetime('2021-06-03T01:48:14Z'))
{
  "results": [{
      "month(datetime('2021-06-03T01:48:14Z'))": 6
    }]
}
```

Di sini, `datetime` dinormalisasi ke UTC, jadi + 08:00 bergeser kembali ke 31 Mei.

```
RETURN month(datetime('2021-06-01T00:00:00+08:00'))
{
  "results": [{
      "month(datetime('2021-06-01T00:00:00+08:00'))": 5
    }]
```

```
}
```

`date`Masukan hanya tersedia di Neptunus Analytics:

```
RETURN month(date('2021-06-03Z'))
{
  "results": [{
      "month(date('2021-06-03Z'))": 6
    }]
}
```

Zona waktu `date` mempertahankan, menjaga 1 Juni.

```
RETURN month(date('2021-06-01+08:00'))
{
  "results": [{
      "month(date('2021-06-01+08:00'))": 6
    }]
}
```

tahun

`year(temporal :: (datetime | date)) :: (LONG)`

Mengembalikan tahun dari `date` nilai `datetime` atau. Untuk`datetime`: nilai dinormalisasi ke UTC berdasarkan input sebelum mengekstraksi tahun. Untuk`date`: tahun diekstraksi berdasarkan zona waktu.

`datetime`Masukan tersedia di Database Neptunus dan Neptunus Analytics:

```
RETURN year(datetime('2021-06-03T01:48:14Z'))
{
  "results": [{
      "year(datetime('2021-06-03T01:48:14Z'))": 2021
    }]
}
```

Di sini, `datetime` dinormalisasi ke UTC, jadi + 08:00 bergeser kembali ke 31 Desember 2020.

```
RETURN year(datetime('2021-01-01T00:00:00+08:00'))
```

```
{
  "results": [{
      "year(datetime('2021-01-01T00:00:00+08:00'))": 2020
    }]
}
```

`date`Masukan hanya tersedia di Neptunus Analytics:

```
RETURN year(date('2021-06-03Z'))
{
  "results": [{
      "year(date('2021-06-03Z'))": 2021
    }]
}
```

Zona waktu `date` mempertahankan, menjaga Juni 2021.

```
RETURN year(date('2021-01-01+08:00'))
{
  "results": [{
      "year(date('2021-01-01+08:00'))": 2021
    }]
}
```

Fungsi Neptunus OpenCypher

Tersedia dalam Database Neptunus 1.4.6.0 dan lebih tinggi, dan Neptunus Analytics.

mengurangi ()

Mengurangi proses secara berurutan setiap elemen daftar dengan menggabungkannya dengan total berjalan atau 'akumulator.' Dimulai dengan nilai awal, ia memperbarui akumulator setelah setiap operasi dan menggunakan nilai yang diperbarui dalam iterasi berikutnya.

`for i in (0, ..., n) acc = acc X list[I], where X denotes any binary operator`

Setelah semua elemen diproses, ia mengembalikan hasil akumulasi akhir.

Struktur reduce () yang khas adalah - `reduce(accumulator = initial , variable IN list | expression)`

Spesifikasi jenis:

`- initial: starting value for the accumulator :: (Long | FLOAT | STRING | LIST? OF (STRING, LONG, FLOAT)) - list: the input list :: LIST OF T where T matches initial type - variable :: represents each element in the input list - expression :: Only supports '+' and '*' operator - return :: Same type as initial`

Pembatasan:

Saat ini, `reduce()` ekspresi hanya mendukung:

- Perkalian Numerik
- Penambahan Numerik
- Penggabungan String
- Daftar Penggabungan

Mereka diwakili oleh * operator + atau. Ekspresi harus berupa ekspresi biner seperti yang ditentukan di bawah ini - `expression pattern: accumulator + any variable or accumulator * any variable`

Penanganan luapan:

Neptunus mendeteksi luapan numerik selama evaluasi dan merespons `reduce()` secara berbeda berdasarkan tipe data:

```
LONG (signed#64-bit)
--------------------
• Valid range: -9#223#372#036#854#775#808#…#9#223#372#036#854#775#807
• If any intermediate or final value falls outside this range,
  Neptune aborts the query with long overflow error message.

FLOAT (IEEE-754 double)
-----------------------
• Largest finite value ≈#1.79#×#10^308
• Larger results overflow to INF
  Once `INF` is produced, it propagates through the remainder
  of the reduction.
```

Contoh:

Lihat contoh berikut untuk fungsi reduce ().

```
1. Long Addition:
RETURN reduce(sum = 0, n IN [1, 2, 3] | sum + n)
{
  "results": [{
      "reduce(sum = 0, n IN [1, 2, 3] | sum + n)": 6
    }]
}

2. String Concatenation:
RETURN reduce(str = "", x IN ["A", "B", "C"] | str + x)
{
  "results": [{
      "reduce(str = "", x IN ["A", "B", "C"] | str + x)": "ABC"
    }]
}

3. List Combination:
RETURN reduce(lst = [], x IN [1, 2, 3] | lst + x)
{
  "results": [{
      "reduce(lst = [], x IN [1, 2, 3] | lst + x)": [1, 2, 3]
    }]
}

4. Float Addition:
RETURN reduce(total = 0.0, x IN [1.5, 2.5, 3.5] | total + x)
{
  "results": [{
      "reduce(total = 0.0, x IN [1.5, 2.5, 3.5] | total + x)": 7.5
    }]
}

5. Long Multiplication:
RETURN reduce(product = 1, n IN [1, 2, 3] | product * n)
{
  "results": [{
      "reduce(product = 0, n IN [1, 2, 3] | product * n)": 6
    }]
}

6. Float Multiplication:
```

```
RETURN reduce(product = 1.0, n IN [1.5, 2.5, 3.5] | product * n)
{
  "results": [{
      "reduce(product = 1.0, n IN [1.5, 2.5, 3.5] | product * n)": 13.125
    }]
}

7. Long Overflow (Exception):
RETURN reduce(s = 9223372036854775807, x IN [2, 3] | s * x) AS result
{
"results": [{
    "reduce(s = 9223372036854775807, x IN [2, 3] | s * x) AS result": long overflow
    }]
}

8. Float Overflow:
RETURN reduce(s = 9.0e307, x IN [8.0e307, 1.0e307] | s + x) AS result
{
"results": [{
    "reduce(s = 9.0e307, x IN [8.0e307, 1.0e307] | s + x) AS result": INF
    }]
}
```

neptune.read ()

Neptunus mendukung `neptune.read` prosedur untuk membaca `CALL` data dari Amazon S3 dan kemudian menjalankan kueri OpenCypher (baca, sisipkan, perbarui) menggunakan data. Prosedur menghasilkan setiap baris dalam file sebagai baris variabel hasil yang dideklarasikan. Ini menggunakan kredensyal IAM dari penelepon untuk mengakses data di Amazon S3. Lihat Mengelola izin untuk neptune.read () untuk mengatur izin. AWS Wilayah bucket Amazon S3 harus berada di wilayah yang sama di mana instance berada. Saat ini, pembacaan lintas wilayah tidak didukung.

Sintaksis

```
CALL neptune.read(
  {
    source: "string",
    format: "parquet/csv",
    concurrency: 10
  }
)
YIELD row
```

```
...
```

Masukan

- source (wajib) - Amazon S3 URI ke satu objek. Awalan Amazon S3 ke beberapa objek tidak didukung.
- format (wajib) - `parquet` dan `csv` didukung.
  - Rincian lebih lanjut tentang format Parket yang didukung dapat ditemukan diJenis kolom Parket yang didukung.
  - Untuk informasi selengkapnya tentang format csv yang didukung, lihatMuat format untuk data OpenCypher.
- konkurensi (opsional) - Jenis: 0 atau bilangan bulat yang lebih besar. Default: 0. Menentukan jumlah thread yang akan digunakan untuk membaca file. Jika nilainya 0, jumlah maksimum utas yang diizinkan oleh sumber daya akan digunakan. Untuk Parket, disarankan untuk diatur ke sejumlah grup baris.

Output

Neptune.read mengembalikan:

- baris - Jenis: Peta
  - Setiap baris dalam file, di mana kuncinya adalah kolom dan nilainya adalah data yang ditemukan di setiap kolom.
  - Anda dapat mengakses data setiap kolom seperti properti access (`row.col`).

Praktik terbaik untuk neptune.read ()

Operasi pembacaan Neptunus S3 bisa intensif memori. Harap gunakan jenis instans yang cocok untuk beban kerja produksi seperti yang diuraikan dalam Memilih jenis instans untuk Amazon Neptunus.

Penggunaan memori dan kinerja `neptune.read()` permintaan dipengaruhi oleh berbagai faktor seperti ukuran file, jumlah kolom, jumlah baris, dan format file. Bergantung pada struktur, file kecil (misalnya, file CSV 100MB atau di bawahnya, file Parket 20MB atau di bawahnya) dapat bekerja dengan andal pada sebagian besar jenis instance yang sesuai dengan produksi, sedangkan file yang lebih besar mungkin memerlukan memori substanal yang tidak dapat disediakan oleh tipe instance yang lebih kecil.

Saat menguji fitur ini, disarankan untuk memulai dengan file kecil dan menskalakan secara bertahap untuk memastikan beban kerja baca Anda dapat diakomodasi oleh ukuran instans Anda. Jika Anda melihat `neptune.read()` permintaan yang mengarah ke out-of-memory pengecualian atau instans dimulai ulang, pertimbangkan untuk membagi file Anda menjadi potongan yang lebih kecil, mengurangi kompleksitas file, atau meningkatkan ke jenis instans yang lebih besar.

Contoh kueri menggunakan parket

Contoh query berikut mengembalikan jumlah baris dalam file Parket tertentu:

```
CALL neptune.read(
  {
    source: "<s3 path>",
    format: "parquet"
  }
)
YIELD row
RETURN count(row)
```

Anda dapat menjalankan contoh query menggunakan `execute-open-cypher-query` operasi di AWS CLI dengan mengeksekusi kode berikut:

```
aws neptunedata execute-open-cypher-query \
--open-cypher-query "CALL neptune.read({source: '<s3 path>', format: 'parquet'}) YIELD
 row RETURN count(row)" \
--endpoint-url https://my-cluster-name.cluster-abcdefgh1234.us-
east-1.neptune.amazonaws.com:8182
```

Kueri bisa fleksibel dalam apa yang dilakukannya dengan baris yang dibaca dari file Parket. Misalnya, kueri berikut membuat simpul dengan bidang yang disetel ke data yang ditemukan di file Parket:

```
CALL neptune.read(
  {
    source: "<s3 path>",
    format: "parquet"
  }
)
YIELD row
CREATE (n {someField: row.someCol})
RETURN n
```

> **Warning**
>
> Tidak dianggap praktik yang baik untuk menggunakan klausa penghasil hasil besar seperti `MATCH(n)` sebelum klausa. `CALL` Ini akan mengarah pada kueri yang berjalan lama, karena produk silang antara solusi yang masuk dari klausa sebelumnya dan baris yang dibaca oleh neptune.read. Disarankan untuk memulai kueri dengan `CALL` neptune.read.

Jenis kolom Parket yang didukung

Jenis Data Parket:

- NULL
- BOOLEAN
- FLOAT
- DOUBLE
- STRING
- INTEGER YANG DITANDATANGANI: UINT8,, UINT16, UINT32 UINT64
- MAP: Hanya mendukung satu tingkat. Tidak mendukung bersarang.
- DAFTAR: Hanya mendukung satu tingkat. Tidak mendukung bersarang.

Tipe data khusus Neptunus:

Berbeda dengan header kolom properti dari format CSV, header kolom properti dari format Parket hanya perlu memiliki nama properti, jadi tidak perlu memiliki nama tipe atau kardinalitas.

Namun, ada beberapa jenis kolom khusus dalam format Parket yang memerlukan anotasi dalam metadata, termasuk tipe Any, tipe Date, tipe DateTime, dan jenis Geometri. Objek berikut adalah contoh anotasi metadata yang diperlukan untuk file yang berisi kolom jenis khusus ini:

```
"metadata": {
    "anyTypeColumns": ["UserCol1"],
    "dateTypeColumns": ["UserCol2"],
    "dateTimeTypeColumns": ["UserCol3"],
    "geometryTypeColumns": ["UserCol4"]
}
```

Di bawah ini adalah detail tentang muatan yang diharapkan terkait dengan jenis ini:

- Tipe kolom Any didukung di kolom pengguna. Tipe Any adalah jenis “gula sintaksis” untuk semua jenis lain yang kami dukung. Ini sangat berguna jika kolom pengguna memiliki beberapa jenis di dalamnya. Payload dari nilai tipe Any adalah daftar string json sebagai berikut:`{"value": "10", "type": "Int"};{"value": "1.0", "type": "Float"}`, yang memiliki bidang nilai dan bidang tipe di setiap string json individu. Nilai kardinalitas kolom Any diatur, artinya kolom dapat menerima beberapa nilai.
  - Neptunus mendukung jenis berikut dalam tipe Any: Bool (atau Boolean), Byte, Short, Int, Long,,,,, Float, Double UnsignedByte UnsignedShort, Date UnsignedInt UnsignedLong, DateTime, String, dan Geometry.
  - Jenis vektor tidak didukung dalam Tipe apa pun.
  - Bersarang Tipe apa pun tidak didukung. Misalnya, `{"value": {"value": "10", "type": "Int"}, "type": "Any"}`.
- Kolom tipe Tanggal dan Datetime didukung di kolom pengguna. Muatan kolom ini harus disediakan sebagai string mengikuti format XSD atau salah satu format di bawah ini:
  - YYYY-MM-DD
  - YYYY-MM-DDTHH: mm
  - YYYY-MM-DDTHH: MM: SS
  - YYYY-MM-DDTHH: mm: SSZ
  - YYYY-MM-DDTHH: MM: SS.SSSZ
  - YYYY-MM-DDTHH: mm: SS [+|-] hhmm
  - YYYY-MM-DDTHH: mm: SS.sss [+|-] hhmm
- Jenis kolom Geometri didukung di kolom pengguna. Muatan kolom ini hanya boleh berisi geometri primitif tipe Point, disediakan sebagai string dalam format teks terkenal (WKT). Misalnya, POINT (30 10) akan menjadi nilai Geometri yang valid.

Contoh keluaran parket

Diberikan file Parket seperti ini:

```
<s3 path>

Parquet Type:
```

```
    int8     int16       int32            int64              float      double
 string
+--------+---------+------------+--------------------+-----------+-----------
+----------+
|   Byte |   Short |       Int   |               Long  |     Float  |    Double  |
 String   |
|--------+---------+------------+--------------------+-----------+-----------
+----------|
|   -128 |  -32768 | -2147483648 | -9223372036854775808 |    1.23456 |    1.23457 |
 first    |
|    127 |   32767 |  2147483647 |  9223372036854775807 |  nan       |  nan       |
 second   |
|      0 |       0 |           0 |                    0 | -inf       | -inf       |
 third    |
|      0 |       0 |           0 |                    0 |  inf       |  inf       |
 fourth   |
+--------+---------+------------+--------------------+-----------+-----------
+----------+
```

Berikut adalah contoh output yang dikembalikan oleh neptune.read menggunakan query berikut:

```
aws neptunedata execute-open-cypher-query \
--open-cypher-query "CALL neptune.read({source: '<s3 path>', format: 'parquet'}) YIELD
 row RETURN row" \
--endpoint-url https://my-cluster-name.cluster-abcdefgh1234.us-
east-1.neptune.amazonaws.com:8182
```

```
{
 "results": [{
 "row": {
 "Float": 1.23456,
 "Byte": -128,
 "Int": -2147483648,
 "Long": -9223372036854775808,
 "String": "first",
 "Short": -32768,
 "Double": 1.2345678899999999
 }
 }, {
 "row": {
 "Float": "NaN",
 "Byte": 127,
 "Int": 2147483647,
```

```
 "Long": 9223372036854775807,
 "String": "second",
 "Short": 32767,
 "Double": "NaN"
 }
 }, {
 "row": {
 "Float": "-INF",
 "Byte": 0,
 "Int": 0,
 "Long": 0,
 "String": "third",
 "Short": 0,
 "Double": "-INF"
 }
 }, {
 "row": {
 "Float": "INF",
 "Byte": 0,
 "Int": 0,
 "Long": 0,
 "String": "fourth",
 "Short": 0,
 "Double": "INF"
 }
 }]
}
```

Saat ini, tidak ada cara untuk mengatur node atau label tepi ke bidang data yang berasal dari file Parket. Disarankan agar Anda mempartisi kueri menjadi beberapa kueri, satu untuk setiap label/jenis.

```
CALL neptune.read({source: '<s3 path>', format: 'parquet'})
 YIELD row
WHERE row.`~label` = 'airport'
CREATE (n:airport)

CALL neptune.read({source: '<s3 path>', format: 'parquet'})
YIELD row
WHERE row.`~label` = 'country'
CREATE (n:country)
```

Contoh kueri menggunakan CSV

Dalam contoh ini, kueri mengembalikan jumlah baris dalam file CSV yang diberikan:

```
CALL neptune.read(
  {
    source: "<s3 path>",
    format: "csv"
  }
)
YIELD row
RETURN count(row)
```

Anda dapat menjalankan contoh query menggunakan execute-open-cypher-query operasi di AWS CLI dengan mengeksekusi kode berikut:

```
aws neptunedata execute-open-cypher-query \
--open-cypher-query "CALL neptune.read({source: '<s3 path>', format: 'csv'}) YIELD row
 RETURN count(row)" \
--endpoint-url https://my-cluster-name.cluster-abcdefgh1234.us-
east-1.neptune.amazonaws.com:8182
```

Kueri bisa fleksibel dalam apa yang dilakukannya dengan baris yang dibaca dari file CSV. Misalnya, kueri berikut membuat simpul dengan bidang yang disetel ke data dari file CSV:

```
CALL neptune.read(
  {
    source: "<s3 path>",
    format: "csv"
  }
)
YIELD row
CREATE (n {someField: row.someCol})
RETURN n
```

> **Warning**
>
> Ini tidak dianggap praktik yang baik menggunakan klausa penghasil hasil besar seperti MATCH (n) sebelum klausa CALL. Ini akan mengarah pada kueri yang berjalan lama karena

produk silang antara solusi yang masuk dari klausa sebelumnya dan baris yang dibaca oleh neptune.read. Disarankan untuk memulai kueri dengan CALL neptune.read.

Header kolom properti

Anda dapat menentukan kolom (`:`) untuk properti dengan menggunakan sintaks berikut. Nama jenis tidak peka dengan huruf besar/kecil. Jika titik dua muncul dalam nama properti, itu harus diloloskan dengan mendahuluinya dengan garis miring terbalik:. `\:`

```
propertyname:type
```

Note

- Spasi, koma, carriage return, dan karakter baris baru tidak diperbolehkan di header kolom, sehingga nama properti tidak dapat menyertakan karakter ini.
- Anda dapat menentukan kolom untuk jenis array dengan menambahkan `[]` ke jenisnya:

  ```
  propertyname:type[]
  ```

- Properti edge hanya dapat memiliki satu nilai dan akan menyebabkan kesalahan jika jenis array yang ditentukan atau nilai kedua ditentukan. Contoh berikut menunjukkan header kolom untuk properti bernama umur tipe Int:

  ```
  age:Int
  ```

Setiap baris dalam file akan perlu memiliki integer dalam posisi itu atau dibiarkan kosong. Array string diperbolehkan, tetapi string dalam array tidak dapat menyertakan karakter titik koma () kecuali jika diloloskan menggunakan garis miring terbalik (`;`). `\;`

Jenis kolom CSV yang didukung

- BOOL (atau BOOLEAN) - Nilai yang diizinkan: benar, salah. Menunjukkan bidang Boolean. Nilai apa pun selain true akan diperlakukan sebagai false.

- FLOAT - Rentang: 32-bit IEEE 754 floating point termasuk Infinity, INF, -Infinity, -INF dan NaN (). not-a-number
- DOUBLE - Rentang: 64-bit IEEE 754 floating point termasuk Infinity, INF, -Infinity, -INF dan NaN (). not-a-number
- TALI -
  - Tanda kutip adalah opsional. Karakter koma, baris baru, dan carriage return secara otomatis lolos jika disertakan dalam string yang dikelilingi oleh tanda kutip ganda (“). Contoh: “Halo, Dunia”.
  - Untuk menyertakan tanda kutip dalam string yang dikutip, Anda dapat menghindari tanda kutip dengan menggunakan dua berturut-turut: Contoh: “Halo “" Dunia """”.
  - Array string diperbolehkan, tetapi string dalam array tidak dapat menyertakan karakter titik koma (;) kecuali jika diloloskan menggunakan garis miring terbalik (\;).
  - Jika Anda ingin mengapit string dalam array dengan tanda kutip, Anda harus mengapit keseluruhan array dengan satu set tanda kutip. Contoh: “String satu; String 2; String 3".
- DATE, DATETIME - Nilai datetime dapat diberikan dalam format XSD, atau salah satu format berikut:
  - YYYY-MM-DD
  - YYYY-MM-DDTHH: mm
  - YYYY-MM-DDTHH: MM: SS
  - YYYY-MM-DDTHH: mm: SSZ
  - YYYY-MM-DDTHH: MM: SS.SSSZ
  - YYYY-MM-DDTHH: mm: SS [+|-] hhmm
  - YYYY-MM-DDTHH: mm: SS.sss [+|-] hhmm
- INTEGER YANG DITANDATANGANI -
  - Byte: -128 hingga 127
  - Pendek: -32768 sampai 32767
  - Int: -2^31 hingga 2 ^ 31-1
  - Panjang: -2 ^ 63 hingga 2 ^ 63-1

Jenis kolom khusus Neptunus:

- Tipe kolom Any didukung di kolom pengguna. Tipe Any adalah jenis “gula sintaksis” untuk semua jenis lain yang kami dukung. Ini sangat berguna jika kolom pengguna memiliki beberapa jenis di

dalamnya. Payload dari nilai tipe Any adalah daftar string json sebagai berikut:`{"value": "10", "type": "Int"};{"value": "1.0", "type": "Float"}`, yang memiliki bidang nilai dan bidang tipe di setiap string json individu. Header kolom dari tipe Any adalah PropertyName:Any. Nilai kardinalitas kolom Any diatur, artinya kolom dapat menerima beberapa nilai.

  - Neptunus mendukung jenis berikut dalam tipe Any: Bool (atau Boolean), Byte, Short, Int, Long,,,,, Float, Double UnsignedByte UnsignedShort, Date UnsignedInt UnsignedLong, DateTime, String, dan Geometry.
  - Jenis vektor tidak didukung dalam Tipe apa pun.
  - Bersarang Tipe apa pun tidak didukung. Misalnya, `{"value": {"value": "10", "type": "Int"}, "type": "Any"}`.
- Jenis kolom Geometri didukung di kolom pengguna. Muatan kolom ini hanya boleh berisi geometri primitif tipe Point, disediakan sebagai string dalam format teks terkenal (WKT). Misalnya, POINT (30 10) akan menjadi nilai Geometri yang valid.

Contoh keluaran CSV

Diberikan file CSV berikut:

```
<s3 path>
colA:byte,colB:short,colC:int,colD:long,colE:float,colF:double,colG:string
-128,-32768,-2147483648,-9223372036854775808,1.23456,1.23457,first
127,32767,2147483647,9223372036854775807,nan,nan,second
0,0,0,0,-inf,-inf,third
0,0,0,0,inf,inf,fourth
```

Contoh ini menunjukkan output yang dikembalikan oleh neptune.read menggunakan query berikut:

```
aws neptunedata execute-open-cypher-query \
--open-cypher-query "CALL neptune.read({source: '<s3 path>', format: 'csv'}) YIELD row
 RETURN row" \
--endpoint-url https://my-cluster-name.cluster-abcdefgh1234.us-
east-1.neptune.amazonaws.com:8182
```

```
{
  "results": [{
      "row": {
        "colD": -9223372036854775808,
        "colC": -2147483648,
```

```
        "colE": 1.23456,
        "colB": -32768,
        "colF": 1.2345699999999999,
        "colG": "first",
        "colA": -128
      }
    }, {
      "row": {
        "colD": 9223372036854775807,
        "colC": 2147483647,
        "colE": "NaN",
        "colB": 32767,
        "colF": "NaN",
        "colG": "second",
        "colA": 127
      }
    }, {
      "row": {
        "colD": 0,
        "colC": 0,
        "colE": "-INF",
        "colB": 0,
        "colF": "-INF",
        "colG": "third",
        "colA": 0
      }
    }, {
      "row": {
        "colD": 0,
        "colC": 0,
        "colE": "INF",
        "colB": 0,
        "colF": "INF",
        "colG": "fourth",
        "colA": 0
      }
    }]
}
```

Saat ini, tidak ada cara untuk mengatur label node atau edge ke bidang data yang berasal dari file CSV. Disarankan agar Anda mempartisi kueri menjadi beberapa kueri, satu untuk setiap label/jenis.

```
CALL neptune.read({source: '<s3 path>', format: 'csv'})
```

```
 YIELD row
WHERE row.`~label` = 'airport'
CREATE (n:airport)

CALL neptune.read({source: '<s3 path>', format: 'csv'})
YIELD row
WHERE row.`~label` = 'country'
CREATE (n:country)
```

Mengelola izin untuk neptune.read ()

Kebijakan IAM yang Diperlukan

Untuk menjalankan kueri OpenCypher yang digunakan`neptune.read()`, Anda harus memiliki izin yang sesuai untuk mengakses data dalam database Neptunus Anda. Kueri hanya-baca memerlukan tindakan. `ReadDataViaQuery` Kueri yang memodifikasi data memerlukan `WriteDataViaQuery` penyisipan atau penghapusan`DeleteDataViaQuery`. Contoh di bawah ini memberikan ketiga tindakan pada cluster yang ditentukan.

Selain itu, Anda memerlukan izin untuk mengakses bucket S3 yang berisi file data Anda. Pernyataan kebijakan NepTunes3Access memberikan izin S3 yang diperlukan:

- **`s3:ListBucket`**: Diperlukan untuk memverifikasi keberadaan bucket dan daftar konten.
- **`s3:GetObject`**: Diperlukan untuk mengakses objek yang ditentukan sehingga kontennya dapat dibaca untuk integrasi ke dalam kueri OpenCypher.

Jika bucket S3 Anda menggunakan enkripsi sisi server AWS KMS, Anda juga harus memberikan izin KMS. KMSAccess Pernyataan kebijakan Neptunes3 memungkinkan Neptunus untuk mendekripsi data dan menghasilkan kunci data saat mengakses objek S3 terenkripsi. Kondisi ini membatasi operasi KMS untuk permintaan yang berasal dari layanan S3 dan RDS di wilayah Anda.

- **`kms:Decrypt`**: Diperlukan untuk melakukan dekripsi objek terenkripsi sehingga datanya dapat dibaca oleh Neptunus.
- **`kms:GenerateDataKey`**: Juga diperlukan oleh S3 API yang digunakan untuk mengambil objek yang akan dibaca.

```
{
  "Sid": "NeptuneQueryAccess",
```

```
  "Effect": "Allow",
  "Action": [
      "neptune-db:ReadDataViaQuery",
      "neptune-db:WriteDataViaQuery",
      "neptune-db:DeleteDataViaQuery"
  ],
  "Resource": "arn:aws:neptune-db:<REGION>:<AWS_ACCOUNT_ID>:<CLUSTER_RESOURCE_ID>/*"
},
{
  "Sid": "NeptuneS3Access",
  "Effect": "Allow",
  "Action": [
      "s3:ListBucket",
      "s3:GetObject"
  ],
  "Resource": [
      "arn:aws:s3:::neptune-read-bucket",
      "arn:aws:s3:::neptune-read-bucket/*"
  ]
},
{
  "Sid": "NeptuneS3KMSAccess",
  "Effect": "Allow",
  "Action": [
      "kms:Decrypt",
      "kms:GenerateDataKey"
  ],
  "Resource": "arn:aws:kms:<REGION>:<AWS_ACCOUNT_ID>:key/<KEY_ID>",
  "Condition": {
      "StringEquals": {
        "kms:ViaService": [
            "s3.<REGION>.amazonaws.com",
            "rds.<REGION>.amazonaws.com"
        ]
      }
  }
}
```

Prasyarat penting

Izin dan prasyarat ini memastikan integrasi data S3 yang aman dan andal ke dalam kueri OpenCypher, sambil mempertahankan kontrol akses yang tepat dan langkah-langkah perlindungan data.

- Autentikasi IAM: Fitur ini hanya didukung untuk cluster Neptunus dengan otentikasi IAM diaktifkan. Lihat Mengamankan database Amazon Neptunus untuk petunjuk terperinci tentang cara membuat dan menyambung ke kluster berkemampuan autentikasi IAM.
- Titik akhir VPC:
  - Titik akhir VPC tipe Gateway untuk Amazon S3 diperlukan untuk memungkinkan Neptunus berkomunikasi dengan Amazon S3.
  - Untuk menggunakan AWS KMS enkripsi kustom dalam kueri, titik akhir VPC tipe antarmuka untuk diperlukan agar AWS KMS Neptunus dapat berkomunikasi dengannya. AWS KMS
  - Untuk petunjuk mendetail tentang cara mengonfigurasi titik akhir ini, lihat Membuat Titik Akhir VPC Amazon S3.

Data Spasial

Amazon Neptunus sekarang mendukung kueri spasial, memungkinkan Anda untuk menyimpan dan menganalisis data geometris dalam grafik Anda. Meskipun umumnya digunakan untuk lokasi geografis (seperti koordinat pada peta), fitur spasial bekerja dengan data dua dimensi di mana posisi dan kedekatan penting. Gunakan fitur ini untuk menjawab pertanyaan seperti “Toko mana yang berada dalam jarak 5 mil dari pelanggan ini?” , “Temukan semua rute pengiriman yang bersinggungan dengan area layanan ini,” atau “Komponen mana dalam denah lantai ini yang tumpang tindih dengan zona HVAC?” Neptunus mengimplementasikan dukungan spasial menggunakan fungsi Tipe Spasial standar industri yang bekerja dengan titik, poligon, dan bentuk geometris lainnya. Anda dapat menyimpan data spasial sebagai properti pada node dan tepi, kemudian menggunakan fungsi spasial untuk menghitung jarak, memeriksa apakah titik berada dalam batas, atau menemukan wilayah yang tumpang tindih, semua dalam kueri OpenCypher Anda.

Kasus penggunaan umum:

- Aplikasi geografis: Rekomendasi berbasis lokasi, geofencing, perencanaan rute, dan analisis wilayah
- Manajemen fasilitas dan ruang: Tata letak denah lantai, penempatan peralatan, dan cakupan zona
- Topologi jaringan: Pemetaan infrastruktur fisik, area cakupan, dan batas layanan
- Desain dan CAD: Pemosisian komponen, deteksi tabrakan, dan hubungan spasial dalam desain 2D
- Pengembangan game: Penentuan posisi karakter, deteksi tabrakan, dan perhitungan area-of-effect

Implementasi Jenis Spasial di Amazon Neptunus mengikuti arahan ISO/IEC 13249-3:2016, seperti database lainnya. Yang Fungsi Spasial tersedia dalam bahasa query OpenCypher.

Sistem koordinat

Neptunus memiliki satu Spatial Reference Identifier (SRID) untuk seluruh database. Homogenitas sistem koordinat mengurangi kesalahan pengguna dalam query dan meningkatkan kinerja database. Rilis pertama (1.4.7.0) mendukung sistem koordinat Cartesian, juga disebut sebagai SRID 0.

Implementasi Neptunus dari SRID 0 kompatibel dengan nilai bujur dan lintang. Gunakan `ST_DistanceSpheroid` untuk menghitung jarak berdasarkan WGS84 /SRID 4326.

Implementasi saat ini mendukung penyimpanan koordinat 3 dimensi. Fungsi Spasial saat ini hanya mendukung penggunaan koordinat sumbu x dan y (2 dimensi). Koordinat sumbu z saat ini tidak didukung oleh Fungsi Spasial yang tersedia.

Menyimpan data lokasi

Menyimpan data lokasi pada node dan tepi menggunakan tipe properti Geometry. Buat nilai Geometri dari format Teks Terkenal (WKT), cara standar untuk mewakili bentuk geografis sebagai teks. Misalnya, untuk menyimpan lokasi titik:

```
CREATE (n:airport {code: 'ATL', location: ST_GeomFromText('POINT (-84.4281 33.6367)')})
```

Saat bekerja dengan koordinat geografis, argumen pertama (x) mewakili garis bujur dan argumen kedua (y) mewakili garis lintang. Ini mengikuti urutan koordinat standar yang digunakan dalam database spasial dan standar ISO 19125.

> **Note**
>
> Neptunus sekarang mendukung tipe data baru yang disebut “Geometri”. Properti Geometri dari node atau edge dapat dibuat dari string WKT menggunakan fungsi. `ST_GeomFromText` Neptunus akan secara otomatis menyimpan data Poin dalam indeks spasial khusus untuk meningkatkan kinerja fungsi Tipe Spasial. Misalnya, `ST_Contains` digunakan untuk menemukan titik-titik dalam poligon dipercepat oleh indeks spasial khusus.
> Halaman Wikipedia untuk representasi Teks Terkenal dari geometri

Memuat data spasial secara massal

Saat memuat data secara massal, tentukan jenis Geometri di header CSV Anda. Neptunus akan mengurai string WKT dan membuat properti Geometri yang sesuai:

```
:ID,:LABEL,code:String,city:String,location:Geometry
21,airport,ATL,Atlanta,POINT (-84.42810059 33.63669968)
32,airport,ANC,Anchorage,POINT (-149.9960022 61.17440033)
43,airport,AUS,Austin,POINT (-97.66989899 30.19449997)
```

Untuk detail format CSV lengkap, lihat Format beban massal OpenCypher.

Meminta data spasial

Contoh kueri berikut menggunakan dataset rute udara untuk menunjukkan cara menggunakan Fungsi Spasial di Neptunus.

Jika data Anda memiliki properti lintang dan bujur yang terpisah alih-alih properti Geometri, Anda dapat mengonversinya menjadi titik pada waktu kueri. Temukan 10 bandara terdekat ke lokasi tertentu:

```
MATCH (a:airport)
WITH a, ST_GeomFromText('POINT (' + a.lon + ' ' + a.lat + ')') AS airportLocation
WITH a, airportLocation, ST_Distance(ST_GeomFromText('POINT (-84.4281 33.6367)'),
 airportLocation) AS distance
WHERE distance IS NOT NULL
RETURN a.code, a.city, distance
ORDER BY distance ASC
LIMIT 10
```

Jika Anda sudah memiliki lokasi yang disimpan`ST_Point`, Anda dapat menggunakan nilai lokasi tersebut secara langsung:

1. Mengatur properti

   ```
   MATCH (a:airport)
   SET a.location = ST_GeomFromText('POINT (' + a.lon + ' ' + a.lat + ')')
   ```

2. Kueri menggunakan ST_Distance:

   ```
   MATCH (a:airport)
   ```

```
WHERE a.location IS NOT NULL
WITH a, ST_Distance(ST_GeomFromText('POINT (-84.4281 33.6367)'), a.location) AS
 distance
RETURN a.code, a.city, distance
ORDER BY distance ASC
LIMIT 10
```

Menggunakan driver Bolt

Sebagian besar metode kueri mengembalikan nilai Geometri sebagai string WKT, yang dapat dibaca manusia. Jika Anda menggunakan driver Bolt, nilai Geometri dikembalikan dalam format WKB (Biner Terkenal) untuk efisiensi. Mengkonversi WKB ke objek Geometri dalam aplikasi Anda:

```
try (Session session = driver.session()) {
    Result result = session.run("MATCH (n:airport {code: 'ATL'}) RETURN n.location as
 geom");

    Record record = result.single();
    byte[] wkbBytes = record.get("geom").asByteArray();

    // Convert WKB to Geometry object using JTS library
    WKBReader wkbReader = new WKBReader();
    Geometry geom = wkbReader.read(wkbBytes);
}
```

Fungsi Spasial

Fungsi spasial berikut tersedia di Neptunus OpenCypher untuk bekerja dengan tipe data geometri:

- ST_titik
- ST_ GeomFromText
- ST_ AsText
- ST_ GeometryType
- ST_sama
- ST_Berisi
- ST_berpotongan
- ST_Jarak
- ST_ DistanceSpheroid

- ST_amplop
- ST_Buffer

ST_titik

ST_point mengembalikan titik dari nilai koordinat masukan.

Sintaksis

```
ST_Point(x, y, z)
```

Argumen

- x- Nilai tipe data PRESISI GANDA yang mewakili koordinat pertama.
- y- Nilai tipe data PRESISI GANDA yang mewakili koordinat kedua.
- z- (opsional)

Urutan koordinat

Saat bekerja dengan koordinat geografis, argumen pertama (**x**) mewakili garis bujur dan argumen kedua (**y**) mewakili garis lintang. Ini mengikuti urutan koordinat standar yang digunakan dalam database spasial dan standar ISO 19125.

```
// Correct: longitude first, latitude second
ST_Point(-84.4281, 33.6367)  // Atlanta airport

// Incorrect: latitude first, longitude second
ST_Point(33.6367, -84.4281)  // This will return NaN in distance calculations
```

Rentang koordinat yang valid

Untuk data geografis, pastikan koordinat berada dalam rentang yang valid:

- Bujur (x): -180 hingga 180
- Lintang (y): -90 hingga 90

Koordinat di luar rentang ini akan kembali `NaN` (Bukan Angka) bila digunakan dengan fungsi perhitungan jarak seperti`ST_DistanceSpheroid`.

Jenis pengembalian

GEOMETRI dari titik subtipe

Jika x atau y adalah null, maka null dikembalikan.

Contoh

Berikut ini membangun geometri titik dari koordinat masukan.

```
RETURN ST_Point(5.0, 7.0);
POINT(5 7)
```

ST_ GeomFromText

ST_ GeomFromText membangun objek geometri dari representasi teks terkenal (WKT) dari geometri input.

Sintaksis

```
ST_GeomFromText(wkt_string)
```

Argumen

- `wkt_string`- Nilai tipe data STRING yang merupakan representasi WKT dari geometri.

Jenis pengembalian

GEOMETRY

Jika wkt_string adalah null, maka null dikembalikan.

Jika wkt_string tidak valid, maka a dikembalikan. BadRequestException

Contoh

```
RETURN ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))')
POLYGON((0 0,0 1,1 1,1 0,0 0))
```

ST_ AsText

ST_ AsText mengembalikan representasi teks terkenal (WKT) dari geometri input.

Sintaksis

```
ST_AsText(geo)
```

Argumen

- `geo`- Nilai tipe data GEOMETRI, atau ekspresi yang mengevaluasi GEOMETRI.

Jenis pengembalian

STRING

Jika geo adalah null, maka null dikembalikan.

Jika parameter input bukan Geometri, maka a BadRequestException dikembalikan.

Jika hasilnya lebih besar dari STRING 64-KB, maka kesalahan dikembalikan.

Contoh

```
RETURN ST_AsText(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'))
POLYGON((0 0,0 1,1 1,1 0,0 0))
```

ST_ GeometryType

ST_ GeometryType mengembalikan jenis geometri sebagai string.

Sintaksis

```
ST_GeometryType(geom)
```

Argumen

- `geom`- Nilai tipe data GEOMETRI atau ekspresi yang mengevaluasi tipe GEOMETRI.

Jenis pengembalian

STRING

Jika geom adalah null, maka null dikembalikan.

Jika parameter input bukan Geometri, maka a BadRequestException dikembalikan.

Contoh

```
RETURN ST_GeometryType(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'));
ST_LineString
```

ST_sama

ST_equals mengembalikan true jika proyeksi 2D dari geometri masukan secara topologi sama. Geometri dianggap sama secara topologi jika memiliki himpunan titik yang sama. Dalam geometri yang sama secara topologi, urutan simpul mungkin berbeda sambil mempertahankan kesetaraan ini.

Sintaksis

```
ST_Equals(geom1, geom2)
```

Argumen

- `geom1`- Nilai tipe data GEOMETRI atau ekspresi yang mengevaluasi tipe GEOMETRI.
- `geom2`- Nilai tipe data GEOMETRI atau ekspresi yang mengevaluasi tipe GEOMETRI. Nilai ini dibandingkan dengan geom1 untuk menentukan apakah itu sama dengan geom1.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika geom1 atau geom2 bukan Geometri, maka a dikembalikan. BadRequestException

Contoh

```
RETURN ST_Equals(
    ST_GeomFromText('POLYGON ((0 2,1 1,0 -1,0 2))'),
    ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
false
```

Berikut ini memeriksa apakah kedua garis garis secara geometris sama.

```
RETURN ST_Equals(
    ST_GeomFromText('LINESTRING (1 0, 10 0)'),
```

```
    ST_GeomFromText('LINESTRING(1 0,5 0,10 0)'));
true
```

ST_Berisi

ST_contains mengembalikan true jika proyeksi 2D dari geometri input pertama berisi proyeksi 2D dari geometri input kedua. Geometri A berisi geometri B jika setiap titik di B adalah titik di A, dan interiornya memiliki persimpangan yang tidak kosong. ST_contains (A, B) setara dengan st_Within (B, A).

Sintaksis

```
ST_Contains(geom1, geom2)
```

Argumen

- `geom1`- Nilai tipe GEOMETRI atau ekspresi yang mengevaluasi tipe GEOMETRI.
- `geom2`- Nilai tipe GEOMETRI atau ekspresi yang mengevaluasi tipe GEOMETRI. Nilai ini dibandingkan dengan geom1 untuk menentukan apakah itu terkandung dalam geom1.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika parameter input bukan Geometri, maka a BadRequestException dikembalikan.

Contoh

```
RETURN ST_Contains(
    ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
    ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
false
```

ST_berpotongan

ST_Intersects mengembalikan true jika proyeksi 2D dari dua geometri input memiliki setidaknya satu titik yang sama.

Sintaksis

```
ST_Intersects(geom1, geom2)
```

Argumen

- geom1- Nilai tipe data GEOMETRI atau ekspresi yang mengevaluasi tipe GEOMETRI.
- geom2- Nilai tipe data GEOMETRI atau ekspresi yang mengevaluasi tipe GEOMETRI.

Jenis pengembalian

BOOLEAN

Jika geom1 atau geom2 adalah null, maka null dikembalikan.

Jika parameter input bukan Geometri, maka a BadRequestException dikembalikan.

Contoh

```
RETURN ST_Intersects(
    ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(2 2,2 5,5 5,5 2,2 2))'),
    ST_GeomFromText('MULTIPOINT((4 4),(6 6))'));
true
```

ST_Jarak

Untuk geometri masukan, ST_distance mengembalikan jarak Euclidean minimum antara proyeksi 2D dari dua nilai geometri input.

Sintaksis

```
ST_Distance(geo1, geo2)
```

Argumen

- geo1- Nilai tipe data GEOMETRI, atau ekspresi yang mengevaluasi tipe GEOMETRI.
- geo2- Nilai tipe data GEOMETRI, atau ekspresi yang mengevaluasi GEOMETRI.

Jenis pengembalian

PRESISI GANDA dalam satuan yang sama dengan geometri input.

Jika geo1 atau geo2 adalah null, maka null dikembalikan.

Jika parameter input bukan Geometri, maka a BadRequestException dikembalikan.

Contoh

```
RETURN ST_Distance(
    ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
    ST_GeomFromText('POLYGON((-1 -3,-2 -1,0 -3,-1 -3))'));
1.4142135623731
```

ST_ DistanceSpheroid

Mengembalikan jarak minimum dalam meter antara dua lon/lat geometri. Spheroid adalah WGS84 / SRID 4326.

Sintaksis

```
ST_DistanceSpheroid(geom1, geom2);
```

Argumen

- `geom1`- Nilai tipe data GEOMETRI atau ekspresi yang mengevaluasi tipe GEOMETRI.
- `geom2`- Nilai tipe data GEOMETRI atau ekspresi yang mengevaluasi tipe GEOMETRI.

Jenis pengembalian

FLOAT

Jika geom adalah null, maka null dikembalikan.

Contoh

```
RETURN ST_DistanceSpheroid(
    ST_GeomFromText('POINT(-110 42)'),
    ST_GeomFromText('POINT(-118 38)'))
814278.77
```

ST_amplop

ST_Envelope mengembalikan kotak batas minimum dari geometri masukan, sebagai berikut:

- Jika geometri input kosong, geometri yang dikembalikan akan menjadi POINT EMPTY.
- Jika kotak pembatas minimum dari geometri input merosot ke suatu titik, geometri yang dikembalikan adalah sebuah titik.
- Jika tidak ada yang sebelumnya benar, fungsi mengembalikan counter-clockwise-oriented poligon yang simpulnya adalah sudut kotak pembatas minimum.

Untuk semua input nonempty, fungsi beroperasi pada proyeksi 2D dari geometri input.

Sintaksis

```
ST_Envelope(geom)
```

Argumen

- `geom`- Nilai tipe data GEOMETRI atau ekspresi yang mengevaluasi tipe GEOMETRI.

Jenis pengembalian

GEOMETRY

Jika geom adalah null, maka null dikembalikan.

Contoh

```
RETURN ST_Envelope(ST_GeomFromText("POLYGON ((2 1, 4 3, 6 1, 5 5, 3 4, 2 1))"))
POLYGON ((2 1, 6 1, 6 5, 2 5, 2 1))
```

ST_Buffer

ST_Buffer mengembalikan geometri 2D yang mewakili semua titik yang jaraknya dari geometri input yang diproyeksikan pada bidang XY-Cartesian kurang dari atau sama dengan jarak input.

Sintaksis

```
ST_Buffer(geom, distance, number_of_segments_per_quarter_circle)
```

Argumen

- `geom`- Nilai tipe data GEOMETRI atau ekspresi yang mengevaluasi tipe GEOMETRI.
- `distance`- Nilai tipe data PRESISI GANDA yang mewakili jarak (atau radius) buffer.

- `number_of_segments_per_quarter_circle`- Nilai tipe data INTEGER (harus lebih besar atau sama dengan 0). Nilai ini menentukan jumlah titik untuk mendekati seperempat lingkaran di sekitar setiap simpul geometri input. Nilai negatif default ke nol. Defaultnya adalah 8.

Jenis pengembalian

GEOMETRY

Fungsi ST_Buffer mengembalikan geometri dua dimensi (2D) di bidang XY-Cartesian.

Contoh

```
RETURN ST_Buffer(ST_GeomFromText('LINESTRING (1 2,5 2,5 8)'), 2, 4);
POLYGON ((3 4, 3 8, 3.1522409349774265 8.76536686473018,
         3.585786437626905 9.414213562373096, 4.234633135269821 9.847759065022574,
         5 10, 5.765366864730179 9.847759065022574,
         6.414213562373095 9.414213562373096, 6.847759065022574 8.76536686473018,
         7 8, 7 2, 6.847759065022574 1.2346331352698203,
         6.414213562373095 0.5857864376269051, 5.765366864730179 0.1522409349774265,
         5 0, 1 0, 0.2346331352698193 0.152240934977427,
         -0.4142135623730954 0.5857864376269051,
         -0.8477590650225737 1.2346331352698208, -1 2.0000000000000004,
         -0.8477590650225735 2.7653668647301797,
         -0.4142135623730949 3.414213562373095,
         0.2346331352698206 3.8477590650225735, 1 4, 3 4))
```

Berikut ini mengembalikan buffer dari geometri titik input yang mendekati lingkaran. Karena perintah menentukan 3 sebagai jumlah segmen per seperempat lingkaran, fungsi menggunakan tiga segmen untuk memperkirakan lingkaran seperempat.

```
RETURN ST_Buffer(ST_GeomFromText('POINT (1 1)'), 1.0, 8));
POLYGON ((2 1, 1.9807852804032304 0.8049096779838718,
     1.9238795325112867 0.6173165676349102, 1.8314696123025453 0.4444297669803978,
     1.7071067811865475 0.2928932188134525, 1.5555702330196022 0.1685303876974548,
     1.3826834323650898 0.0761204674887133, 1.1950903220161284 0.0192147195967696,
     1 0, 0.8049096779838718 0.0192147195967696, 0.6173165676349103 0.0761204674887133,
    0.444429766980398 0.1685303876974545, 0.2928932188134525 0.2928932188134524,
     0.1685303876974546 0.4444297669803978, 0.0761204674887133 0.6173165676349102,
     0.0192147195967696 0.8049096779838714, 0 0.9999999999999999,
     0.0192147195967696 1.1950903220161284, 0.0761204674887132 1.3826834323650896,
     0.1685303876974545 1.555570233019602, 0.2928932188134523 1.7071067811865475,
     0.4444297669803978 1.8314696123025453, 0.6173165676349097 1.9238795325112865,
```

```
    0.8049096779838714 1.9807852804032304, 0.9999999999999998 2,
    1.1950903220161284 1.9807852804032304, 1.38268343236509 1.9238795325112865,
    1.5555702330196017 1.8314696123025453, 1.7071067811865475 1.7071067811865477,
    1.8314696123025453 1.5555702330196022, 1.9238795325112865 1.3826834323650905,
    1.9807852804032304 1.1950903220161286, 2 1))
```

Mengakses grafik Neptune dengan SPARQL

SPARQL adalah bahasa kueri untuk Resource Description Framework (RDF), yang merupakan format data grafik yang dirancang untuk web. Amazon Neptune kompatibel dengan SPARQL 1.1. Ini berarti bahwa Anda dapat terhubung ke instans DB Neptune dan mengajukan kueri grafik menggunakan bahasa kueri yang dijelaskan dalam spesifikasi Bahasa Kueri SPARQL 1.1.

Sebuah kueri di SPARQL terdiri dari klausa `SELECT` untuk menentukan variabel yang akan dikembalikan dan klausa `WHERE` untuk menentukan data yang mana yang akan dicocokkan dalam grafik. Jika Anda tidak terbiasa dengan kueri SPARQL, lihat Menulis Kueri Sederhana dalam Bahasa Kueri SPARQL 1.1.

> **Important**
>
> Untuk memuat data, `SPARQL UPDATE INSERT` dapat bekerja dengan baik untuk set data kecil, tetapi jika Anda perlu memuat sejumlah besar data dari sebuah file, lihat Menggunakan pemuat massal Amazon Neptunus untuk menelan data.

Untuk informasi selengkapnya tentang spesifikasi implementasi SPARQL Neptune, lihat Kepatuhan standar SPARQL.

Sebelum Anda mulai, Anda harus memiliki yang berikut:

- Instans DB Neptune. Untuk informasi tentang membuat instans DB Neptune, lihat Membuat cluster Amazon Neptunus.
- Instans Amazon EC2 di virtual private cloud (VPC) yang sama seperti instans DB Neptune Anda.

Topik

- Menggunakan konsol RDF4 J untuk terhubung ke instance DB Neptunus
- Menggunakan RDF4 J Workbench untuk terhubung ke instans DB Neptunus

- Gunakan Java untuk terhubung ke instans DB Neptune
- API HTTP SPARQL
- Petunjuk kueri SPARQL
- Perilaku DESKRIPSI SPARQL sehubungan dengan grafik default
- API status kueri SPARQL
- Pembatalan kueri SPARQL
- Menggunakan Protokol HTTP (GSP) SPARQL 1.1 Graph Store di Amazon Neptunus
- Menganalisis eksekusi kueri Neptune menggunakan explain SPARQL
- Kueri gabungan SPARQL di Neptune menggunakan ekstensi SERVICE

Menggunakan konsol RDF4 J untuk terhubung ke instance DB Neptunus

Konsol RDF4 J memungkinkan Anda bereksperimen dengan grafik dan kueri Resource Description Framework (RDF) di lingkungan REPL (loop). read-eval-print

Anda dapat menambahkan database grafik jarak jauh sebagai repositori dan menanyakannya dari Konsol RDF4 J. Bagian ini memandu Anda melalui konfigurasi Konsol RDF4 J untuk terhubung dari jarak jauh ke instans DB Neptunus.

Untuk terhubung ke Neptunus menggunakan Konsol J RDF4

1. Unduh RDF4 J SDK dari halaman Unduh di situs web RDF4 J.
2. Buka zip file zip RDF4 J SDK.
3. Di terminal, arahkan ke direktori RDF4 J SDK, lalu masukkan perintah berikut untuk menjalankan Konsol RDF4 J:

   ```
   bin/console.sh
   ```

   Anda akan melihat output yang serupa dengan yang berikut:

   ```
   14:11:51.126 [main] DEBUG o.e.r.c.platform.PlatformFactory - os.name = linux
   14:11:51.130 [main] DEBUG o.e.r.c.platform.PlatformFactory - Detected Posix
    platform
   Connected to default data directory
   RDF4J Console 3.6.1
   ```

```
3.6.1
Type 'help' for help.
>
```

Anda sekarang berada di prompt >. Ini adalah prompt umum untuk Konsol RDF4 J. Anda menggunakan prompt ini untuk menyiapkan repositori dan operasi lainnya. Repositori memiliki prompt sendiri untuk menjalankan kueri.

4. Di prompt >, masukkan hal berikut ini untuk membuat repositori SPARQL untuk instans DB Neptune Anda:

```
create sparql
```

5. Konsol RDF4 J meminta Anda untuk nilai untuk variabel yang diperlukan untuk terhubung ke titik akhir SPARQL.

```
Please specify values for the following variables:
```

Tentukan nilai-nilai berikut ini:

Nama variabel	Nilai
Titik akhir kueri SPARQL	`https://`*`your-neptune-endpoint`* `:`*`port`*`/sparql`
Titik akhir pembaruan SPARQL	`https://`*`your-neptune-endpoint`* `:`*`port`*`/sparql`
ID repositori lokal [titik akhir @localhost]	`neptune`
Judul repositori [repositori titik akhir SPARQL @localhost]	`Neptune DB instance`

Untuk informasi tentang menemukan alamat instans DB Neptune Anda, lihat bagian Menghubungkan ke Titik Akhir Amazon Neptune..

Jika operasi berhasil, Anda akan melihat pesan berikut:

```
Repository created
```

6. Di prompt >, masukkan hal berikut untuk menyambung ke instans DB Neptune.

```
open neptune
```

Jika operasi berhasil, Anda akan melihat pesan berikut:

```
Opened repository 'neptune'
```

Anda sekarang berada di prompt `neptune>`. Pada prompt ini, Anda dapat menjalankan kueri terhadap grafil Neptune.

> **Note**
>
> Sekarang setelah Anda menambahkan repositori, saat berikutnya Anda menjalankan `bin/console.sh`, Anda dapat segera menjalankan perintah `open neptune` untuk menyambung ke instans DB Neptune.

7. Pada `neptune>` prompt, masukkan berikut ini untuk menjalankan query SPARQL yang mengembalikan hingga 10 dari triple (subject-predicate-object) dalam grafik dengan menggunakan `?s ?p ?o` kueri dengan batas 10. Untuk mengajukan kueri untuk sesuatu yang lain, gantikan teks setelah perintah `sparql` dengan kueri SPARQL lain.

```
sparql select ?s ?p ?o where {?s ?p ?o} limit 10
```

Menggunakan RDF4 J Workbench untuk terhubung ke instans DB Neptunus

Bagian ini memandu Anda melalui koneksi ke instans Amazon Neptunus DB RDF4 menggunakan J RDF4 Workbench dan J Server. RDF4J Server diperlukan karena bertindak sebagai proxy antara titik akhir Neptunus SPARQL HTTP REST dan J Workbench. RDF4

RDF4J Workbench menyediakan antarmuka yang mudah untuk bereksperimen dengan grafik, termasuk memuat file lokal. Untuk informasi, lihat bagian Tambah dalam dokumentasi RDF4 J.

Prasyarat

Sebelum memulai, lakukan hal berikut:

- Pasang Java 1.8 atau yang lebih baru.
- Instal RDF4 J Server dan RDF4 J Workbench. Untuk selengkapnya, lihat Menginstal RDF4 J Server dan RDF4 J Workbench.

Untuk menggunakan RDF4 J Workbench untuk terhubung ke Neptunus

1. Di browser web, navigasikan ke URL tempat aplikasi web RDF4 J Workbench digunakan. Misalnya, jika Anda menggunakan Apache Tomcat, URL-nya adalah: https: //:8080/rdf4j-workbench/. *ec2_hostname*
2. Jika Anda diminta untuk Connect to RDF4 J Server, verifikasi bahwa RDF4J Server diinstal, berjalan, dan bahwa URL server sudah benar. Kemudian, lanjutkan ke langkah berikutnya.
3. Di panel sebelah kiri, pilih Repositori baru.

   Dalam Repositori baru:

   - Di daftar menurun Jenis, pilih Proksi titik akhir SPARQL.
   - Untuk ID, ketikkan Neptune.
   - Untuk Judul, ketikkan Instans DB Neptune.

   Pilih Berikutnya.
4. Dalam Repositori baru:

   - Untuk URL titik akhir kueri SPARQL, ketikkan `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.
   - Untuk URL titik akhir pembaruan SPARQL, ketikkan `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.

   Untuk informasi tentang menemukan alamat instans DB Neptune Anda, lihat bagian Menghubungkan ke Titik Akhir Amazon Neptune..

   Pilih Buat.

5. Repositori neptune kini muncul dalam daftar repositori. Mungkin perlu beberapa menit sebelum Anda dapat menggunakan repositori baru.
6. Di kolom Id pada tabel, pilih tautan neptune.
7. Di panel sebelah kiri, pilih Kueri.

> **Note**
>
> Jika item menu di bawah Jelajahi dinonaktifkan, Anda mungkin perlu menyambung kembali ke Server RDF4 J dan memilih repositori neptune lagi.
> Anda dapat melakukan hal ini menggunakan tautan [perubahan] di sudut kanan atas.

8. Di bidang kueri, ketik kueri SPARQL berikut, lalu pilih Eksekusi.

```
select ?s ?p ?o where {?s ?p ?o} limit 10
```

Contoh sebelumnya mengembalikan hingga 10 dari triple (subject-predicate-object) dalam grafik dengan menggunakan `?s ?p ?o` query dengan batas 10.

Gunakan Java untuk terhubung ke instans DB Neptune

Bagian ini memandu Anda melalui proses berjalannya sampel Java lengkap yang menyambungkan ke instans DB Amazon Neptune dan melakukan kueri SPARQL.

Ikuti petunjuk ini dari instans Amazon EC2 di virtual private cloud (VPC) yang sama seperti instans Neptune DB Anda.

Untuk menyambung ke Neptune menggunakan Java

1. Install Apache Maven pada instans EC2 Anda. Jika menggunakan Amazon Linux 2023 (lebih disukai), gunakan:

   ```
   sudo dnf update -y
   sudo dnf install maven -y
   ```

   Jika menggunakan Amazon Linux 2, unduh biner terbaru dari https://maven.apache.org/download.cgi:

```
sudo yum remove maven -y
wget https://dlcdn.apache.org/maven/maven-3/ <version>/binaries/apache-maven-
<version>-bin.tar.gz
sudo tar -xzf apache-maven-<version>-bin.tar.gz -C /opt/
sudo ln -sf /opt/apache-maven-<version> /opt/maven
echo 'export MAVEN_HOME=/opt/maven' >> ~/.bashrc
echo 'export PATH=$MAVEN_HOME/bin:$PATH' >> ~/.bashrc
source ~/.bashrc
```

2. Contoh ini diuji hanya dengan Java 8. Masukkan hal berikut ini untuk menginstal Java 8 pada instans EC2 Anda:

```
sudo yum install java-1.8.0-devel
```

3. Masukkan hal berikut untuk mengatur Java 8 sebagai runtime default pada instans EC2 Anda:

```
sudo /usr/sbin/alternatives --config java
```

   Saat diminta, masukkan nomor untuk Java 8.

4. Masukkan hal berikut untuk mengatur Java 8 sebagai compiler default pada instans EC2 Anda:

```
sudo /usr/sbin/alternatives --config javac
```

   Saat diminta, masukkan nomor untuk Java 8.

5. Di direktori baru, buat file `pom.xml`, lalu buka file tersebut dalam editor teks.

6. Salin hal berikut ke file `pom.xml` dan simpan (Anda biasanya dapat menyesuaikan nomor versi ke versi stabil terbaru):

```
<project xmlns="https://maven.apache.org/POM/4.0.0" xmlns:xsi="https://
www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.amazonaws</groupId>
  <artifactId>RDFExample</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>RDFExample</name>
  <url>https://maven.apache.org</url>
```

```
  <dependencies>
    <dependency>
      <groupId>org.eclipse.rdf4j</groupId>
      <artifactId>rdf4j-runtime</artifactId>
      <version>3.6</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>exec-maven-plugin</artifactId>
          <version>1.2.1</version>
          <configuration>
            <mainClass>com.amazonaws.App</mainClass>
          </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

> **Note**
>
> Jika Anda memodifikasi proyek Maven yang ada, dependensi yang diperlukan disorot dalam kode sebelumnya.

7. Untuk membuat subdirektori untuk kode sumber contoh (`src/main/java/com/amazonaws/`), masukkan hal berikut ke baris perintah:

```
mkdir -p src/main/java/com/amazonaws/
```

8. Di direktori `src/main/java/com/amazonaws/`, buat file bernama `App.java`, lalu buka file tersebut dalam editor teks.

9. Salin hal berikut ke dalam file `App.java`. Ganti *`your-neptune-endpoint`* dengan alamat instans DB Neptunus Anda.

> **Note**
>
> Untuk informasi tentang menemukan nama host instans DB Neptune Anda, lihat Menghubungkan ke Titik Akhir Amazon Neptune..

```
package com.amazonaws;

import org.eclipse.rdf4j.repository.Repository;
import org.eclipse.rdf4j.repository.http.HTTPRepository;
import org.eclipse.rdf4j.repository.sparql.SPARQLRepository;

import java.util.List;
import org.eclipse.rdf4j.RDF4JException;
import org.eclipse.rdf4j.repository.RepositoryConnection;
import org.eclipse.rdf4j.query.TupleQuery;
import org.eclipse.rdf4j.query.TupleQueryResult;
import org.eclipse.rdf4j.query.BindingSet;
import org.eclipse.rdf4j.query.QueryLanguage;
import org.eclipse.rdf4j.model.Value;

public class App
{
    public static void main( String[] args )
    {
        String sparqlEndpoint = "https://your-neptune-endpoint:port/sparql";
        Repository repo = new SPARQLRepository(sparqlEndpoint);
        repo.initialize();

        try (RepositoryConnection conn = repo.getConnection()) {
           String queryString = "SELECT ?s ?p ?o WHERE { ?s ?p ?o } limit 10";

           TupleQuery tupleQuery = conn.prepareTupleQuery(QueryLanguage.SPARQL,
 queryString);

           try (TupleQueryResult result = tupleQuery.evaluate()) {
              while (result.hasNext()) {  // iterate over the result
                   BindingSet bindingSet = result.next();
```

```
                    Value s = bindingSet.getValue("s");
                    Value p = bindingSet.getValue("p");
                    Value o = bindingSet.getValue("o");

                    System.out.print(s);
                    System.out.print("\t");
                    System.out.print(p);
                    System.out.print("\t");
                    System.out.println(o);
                }
            }
        }
    }
}
```

10. Gunakan perintah Maven berikut untuk mengonpilasi dan menjalankan sampel:

```
mvn compile exec:java
```

Contoh sebelumnya mengembalikan hingga 10 dari triple (subject-predicate-object) dalam grafik dengan menggunakan `?s ?p ?o` query dengan batas 10. Untuk mengajukan kueri untuk sesuatu yang lain, gantikan kueri tersebut dengan kueri SPARQL lain.

Iterasi hasil dalam contoh mencetak nilai masing-masing variabel yang dkembalikan. Objek `Value` dikonversi menjadi `String` lalu dicetak. Jika Anda mengubah bagian `SELECT` dari kueri, Anda harus memodifikasi kodenya.

API HTTP SPARQL

Permintaan HTTP SPARQL diterima di titik akhir berikut: `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`

Untuk informasi selengkapnya tentang menyambung ke Amazon Neptune dengan SPARQL, lihat Mengakses grafik Neptune dengan SPARQL.

Untuk informasi selengkapnya tentang protokol SPARQL dan bahasa kueri, lihat Protokol SPARQL 1.1 dan spesifikasi Bahasa kueri SPARQL 1.1.

Topik berikut memberikan informasi tentang format serialisasi SPARQL RDF dan cara menggunakan API HTTP SPARQL dengan Neptune.

Daftar Isi

- Menggunakan titik akhir HTTP REST untuk menyambung ke instans DB Neptune
- Header di belakang HTTP opsional untuk respons SPARQL multi-bagian
- Jenis media RDF yang digunakan oleh SPARQL di Neptune
  - Format serialisasi RDF digunakan oleh SPARQL Neptune
  - Format serialisasi hasil SPARQL yang digunakan oleh Neptune SPARQL
  - Jenis Media yang dapat digunakan Neptune untuk mengimpor data RDF
  - Jenis media yang dapat digunakan Neptune untuk mengekspor hasil kueri
- Menggunakan SPARQL UPDATE LOAD untuk mengimpor data ke Neptune
- Menggunakan SPARQL UPDATE UNLOAD untuk menghapus data dari Neptune

Menggunakan titik akhir HTTP REST untuk menyambung ke instans DB Neptune

> **Note**
>
> Neptunus saat ini tidak mendukung HTTP/2 untuk permintaan REST API. Klien harus menggunakan HTTP/1.1 saat menghubungkan ke titik akhir.

Petunjuk berikut memandu Anda menyambungkan ke titik akhir SPARQL menggunakan perintah curl, menyambungkan melalui HTTPS, dan menggunakan sintaks HTTP. Ikuti petunjuk ini dari instans Amazon EC2 di virtual private cloud (VPC) yang sama seperti instans Neptune DB Anda.

Titik akhir HTTP untuk kueri SPARQL ke instans DB Neptune adalah: `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.

> **Note**
>
> Untuk informasi tentang menemukan nama host instans DB Neptune Anda, lihat Menghubungkan ke Titik Akhir Amazon Neptune..

Amazon Neptune menyediakan titik akhir HTTP untuk kueri SPARQL. Antarmuka REST kompatibel dengan SPARQL versi 1.1.

KUERI Menggunakan HTTP POST

Contoh berikut menggunakan curl untuk mengirimkan **QUERY** SPARQL melalui POST HTTP.

```
curl -X POST --data-binary 'query=select ?s ?p ?o where {?s ?p ?o} limit 10'
 https://your-neptune-endpoint:port/sparql
```

Contoh sebelumnya mengembalikan hingga 10 dari triple (subject-predicate-object) dalam grafik dengan menggunakan `?s ?p ?o` query dengan batas 10. Untuk mengajukan kueri untuk sesuatu yang lain, gantikan dengan kueri SPARQL lain.

> **Note**
>
> Jenis media MIME default respon adalah `application/sparql-results+json` untuk kueri `SELECT` dan `ASK`.
> Jenis MIME default respon adalah `application/n-quads` untuk kueri `CONSTRUCT` dan `DESCRIBE`.
> Untuk daftar jenis media yang digunakan oleh Neptune untuk serialisasi, lihat Format serialisasi RDF digunakan oleh SPARQL Neptune.

PEMBARUAN Menggunakan HTTP POST

Contoh berikut menggunakan curl untuk mengirimkan **UPDATE** SPARQL melalui POST HTTP.

```
curl -X POST --data-binary 'update=INSERT DATA { <https://test.com/s> <https://
test.com/p> <https://test.com/o> . }' https://your-neptune-endpoint:port/sparql
```

Contoh sebelumnya menyisipkan tripel berikut ke dalam grafik default SPARQL: `<https://test.com/s> <https://test.com/p> <https://test.com/o>`

Header di belakang HTTP opsional untuk respons SPARQL multi-bagian

Respon HTTP untuk kueri dan pembaruan SPARQL sering dikembalikan dalam lebih dari satu bagian atau potongan. Sulit untuk mendiagnosa kegagalan yang terjadi setelah kueri atau pembaruan mulai mengirimkan potongan ini, terutama karena yang pertama tiba dengan kode status HTTP `200`.

Kecuali Anda secara eksplisit meminta header trailing, Neptune hanya melaporkan kegagalan tersebut dengan menambahkan pesan kesalahan ke badan pesan, yang biasanya rusak.

Untuk memudahkan deteksi dan diagnosis masalah semacam ini, Anda dapat menyertakan header trailer transfer-encoding (TE) (`te: trailers`) dalam permintaan Anda (lihat, misalnya, halaman

MDN tentang header permintaan TE). Melakukan hal ini akan menyebabkan Neptune menyertakan dua bidang header baru dalam header trailing dari potongan respons:

- `X-Neptune-Status` – berisi kode respons diikuti dengan nama pendek. Misalnya, dalam kasus keberhasilan, header trailing akan berupa: `X-Neptune-Status: 200 OK`. Dalam kasus kegagalan, kode respons akan berupa kode kesalahan mesin Neptune, seperti `X-Neptune-Status: 500 TimeLimitExceededException`.
- `X-Neptune-Detail` – kosong untuk permintaan yang berhasil. Dalam kasus kesalahan, ia berisi pesan kesalahan JSON. Karena hanya karakter ASCII yang diperbolehkan dalam nilai header HTTP, string JSON di-enkode dengan URL. Pesan kesalahan juga masih ditambahkan ke badan pesan respons.

Jenis media RDF yang digunakan oleh SPARQL di Neptune

Data Resource Description Framework (RDF) dapat diserialkan dalam berbagai cara, yang sebagian besar dapat dikonsumsi atau dioutput oleh SPARQL:

Format serialisasi RDF digunakan oleh SPARQL Neptune

- RDF/XML – serialisasi XML dari RDF, didefinisikan dalam Sintaks XML RDF 1.1. Jenis media: `application/rdf+xml`. Ekstensi file umum: `.rdf`.
- N–Tripel – Format teks biasa berbasis baris untuk encoding grafik RDF, didefinisikan dalam N-Tripel RDF 1.1. Jenis media: `application/n-triples`, `text/turtle`, atau `text/plain`. Ekstensi file umum: `.nt`.
- N–Quad – Format teks biasa berbasis baris untuk encoding grafik RDF, didefinisikan dalam N-Quad RDF 1.1. Ini adalah perpanjangan dari N-Tripel. Jenis media: `application/n-quads`, atau `text/x-nquads` saat dikodekan dengan US-ASCII 7-bit. Ekstensi file umum: `.nq`.
- Turtle – Sebuah sintaks tekstual untuk RDF yang didefinisikan dalam Turtle RDF 1.1 yang memungkinkan grafik RDF untuk sepenuhnya ditulis dalam bentuk teks yang ringkas dan alami, dengan singkatan untuk pola penggunaan dan tipe data umum. Turtle menyediakan tingkat kompatibilitas dengan format N-Tripel serta sintaks pola tripel SPARQL. Jenis media: `text/turtle`Ekstensi file umum: `.ttl`.
- Turtle – Sebuah sintaks tekstual untuk RDF yang didefinisikan dalam TriG RDF 1.1 yang memungkinkan grafik RDF untuk sepenuhnya ditulis dalam bentuk teks yang ringkas dan alami, dengan singkatan untuk pola penggunaan dan tipe data umum. TriG adalah perpanjangan dari format Turtle. Jenis media: `application/trig`. Ekstensi file umum: `.trig`.

- N3 (Notation3) – Sebuah bahasa pernyataan dan logika yang didefinisikan dalam Notation3 (N3): Sebuah sintaks RDF yang dapat dibaca. N3 memperluas model data RDF dengan menambahkan formula-formula (benar-benar berbentuk grafik), variabel, implikasi logis, dan predikat fungsional, dan menyediakan alternatif sintaks tekstual untuk RDF/XML. Jenis media: `text/n3`. Ekstensi file umum: `.n3`.
- JSON-LD – Sebuah serialisasi data dan format pesan yang didefinisikan dalam JSON-LD 1.0. Jenis media: `application/ld+json`. Ekstensi file umum: `.jsonld`.
- TriX – Sebuah serialisasi RDF dalam XML, didefinisikan dalam TriX: RDF Triple dalam XML. Jenis media: `application/trix`. Ekstensi file umum: `.trix`.
- Hasil JSON SPARQL – Sebuah serialisasi RDF menggunakan Format JSON Hasil Kueri SPARQL 1.1. Jenis media: `application/sparql-results+json`. Ekstensi file umum: `.srj`.
- RDF4J Binary Format — Format biner untuk pengkodean data RDF, didokumentasikan dalam RDF4J Binary RDF Format. Jenis media: `application/x-binary-rdf`.

Format serialisasi hasil SPARQL yang digunakan oleh Neptune SPARQL

- Hasil XML SPARQL – Format XML untuk format hasil variabel binding dan boolean yang disediakan oleh bahasa kueri SPARQL, didefinisikan dalamFormat XML Hasil Kueri SPARQL (Edisi Kedua). Jenis media: `application/sparql-results+xml`. Ekstensi file umum: `.srx`.
- Hasil CSV dan TSV SPARQL – Penggunaan nilai dipisahkan koma dan nilai dipisahkan tab untuk mengekspresikan hasil kueri SPARQL dari kueri `SELECT`, didefinisikan dalam Format CSV dan TSV Hasil Kueri SPARQL 1.1. Jenis media: `text/csv` untuk nilai yang dipisahkan koma, dan `text/tab-separated-values` untuk nilai yang dipisahkan tab. Ekstensi file umum: `.csv` untuk nilai yang dipisahkan koma, dan `.tsv` untuk nilai yang dipisahkan tab.
- Tabel Hasil Biner – Format biner untuk mengkodekan output kueri SPARQL. Jenis media: `application/x-binary-rdf-results-table`.
- Hasil JSON SPARQL – Sebuah serialisasi RDF menggunakan Format JSON Hasil Kueri SPARQL 1.1. Jenis media: `application/sparql-results+json`.

Jenis Media yang dapat digunakan Neptune untuk mengimpor data RDF

Jenis media yang didukung oleh bulk-loader Neptune

- N-Tiga
- N-Quads

- RDF/XML
- Kura-kura

Jenis media yang dapat diimpor SPARQL UPDATE LOAD

- N-Tiga
- N-Quads
- RDF/XML
- Kura-kura
- TriG
- N3
- JSON-LD

Jenis media yang dapat digunakan Neptune untuk mengekspor hasil kueri

Untuk menentukan format output untuk respons kueri SPARQL, kirim header "`Accept: media-type`" dengan permintaan kueri. Contoh:

```
curl -H "Accept: application/nquads" ...
```

Jenis media RDF yang dapat dioutput SPARQL SELECT dari Neptune

- Hasil JSON SPARQL (Ini adalah pengaturan default)
- Hasil XMLSPARQL
- Tabel Hasil Biner (jenis media: `application/x-binary-rdf-results-table`)
- Nilai yang Dipisahkan Koma (CSV)
- Nilai yang Dipisahkan Tab (TSV)

Jenis media RDF yang dapat dioutput SPARQL ASK dari Neptune

- Hasil JSON SPARQL (Ini adalah pengaturan default)
- Hasil XMLSPARQL
- Boolean (jenis media: `text/boolean`, yang berarti “benar” atau “salah”)

Jenis media RDF yang dapat dioutput SPARQL CONSTRUCT dari Neptune

- N-Quad (Ini adalah pengaturan default)
- RDF/XML
- JSON-LD
- N-Tiga Kali Lipat
- Kura-kura
- N3
- TriX
- TriG
- Hasil SPARQL JSON
- RDF4Format RDF Biner J

Jenis media RDF yang dapat dioutput SPARQL DESCRIBE dari Neptune

- N-Quad (Ini adalah pengaturan default)
- RDF/XML
- JSON-LD
- N-Tiga Kali Lipat
- Kura-kura
- N3
- TriX
- TriG
- Hasil SPARQL JSON
- RDF4Format RDF Biner J

Menggunakan SPARQL UPDATE LOAD untuk mengimpor data ke Neptune

Sintaks perintah SPARQL UPDATE LOAD ditentukan dalam Rekomendasi pembaruan SPARQL 1.1:

```
LOAD SILENT (URL of data to be loaded) INTO GRAPH (named graph into which to load the
 data)
```

- **`SILENT`** – (Opsional) Menyebabkan operasi mengembalikan sukses bahkan jika ada kesalahan selama pemrosesan.

  Hal ini dapat berguna ketika transaksi tunggal berisi beberapa pernyataan seperti "`LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;`" dan Anda ingin transaksinya selesai bahkan jika beberapa data jarak jauh tidak dapat diproses.

- *`URL of data to be loaded`*- (Wajib) Menentukan file data jarak jauh yang berisi data yang akan dimuat ke dalam grafik.

  File jarak jauh harus memiliki salah satu ekstensi berikut:

  - `.nt`untuk NTriples.
  - `.nq`untuk NQuads.
  - `.trig` untuk Trig.
  - `.rdf` untuk RDF/XML.
  - `.ttl` untuk Turtle.
  - `.n3` untuk N3.
  - `.jsonld` untuk JSON-LD.

- **`INTO GRAPH`***`(named graph into which to load the data)`*— (Opsional) Menentukan grafik ke mana data harus dimuat.

  Neptune mengasosiasikan setiap tripel dengan grafik bernama. Anda dapat menentukan grafik bernama default menggunakan URI grafik bernama fallback, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`, seperti ini:

```
INTO GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
```

> **Note**
>
> Bila Anda perlu memuat banyak data, sebaiknya gunakan pemuat massal Neptune alih-alih UPDATE LOAD. Untuk informasi lebih lanjut tentang pemuat massal, lihat Menggunakan pemuat massal Amazon Neptunus untuk menelan data.

Anda dapat menggunakan `SPARQL UPDATE LOAD` untuk memuat data langsung dari Amazon S3, atau dari file yang diperoleh dari server web yang di-host sendiri. Sumber daya yang akan dimuat

harus berada di wilayah yang sama seperti server Neptune, dan titik akhir untuk sumber daya harus diizinkan dalam VPC. Untuk informasi tentang cara membuat titik akhir Amazon S3, lihat Membuat VPC Endpoint Amazon S3.

Semua `SPARQL UPDATE LOAD` URIs harus dimulai dengan`https://`. Ini termasuk Amazon S3 URLs.

Berbeda dengan loader massal Neptune, panggilan ke `SPARQL UPDATE LOAD` sepenuhnya transaksional.

Memuat file langsung dari Amazon S3 ke Neptune menggunakan SPARQL UPDATE LOAD

Karena Neptune tidak mengizinkan Anda untuk melewatkan IAM role ke Amazon S3 ketika menggunakan SPARQL UPDATE LOAD, baik bucket Amazon S3 yang bersangkutan harus publik atau Anda harus menggunakan URL Amazon S3 pre-signed dalam kueri LOAD.

Untuk menghasilkan URL yang telah ditandatangani sebelumnya untuk file Amazon S3, Anda dapat menggunakan AWS CLI perintah seperti ini:

```
aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
 data to load)
```

Kemudian Anda dapat menggunakan URL pre-signed yang dihasilkan di dalam perintah `LOAD` Anda:

```
curl https://(a Neptune endpoint URL):8182/sparql \
  --data-urlencode 'update=load (pre-signed URL of the remote Amazon S3 file of data to
 be loaded) \
                          into graph (named graph)'
```

Untuk informasi lebih lanjut, lihat Melakukan Autentikasi Permintaan: Menggunakan Parameter Kueri. Dokumentasi Boto3 menunjukkan cara menggunakan script Python untuk menghasilkan URL pre-signed.

Selain itu, jenis konten file yang akan dimuatkan mesti ditetapkan dengan benar.

1. Atur jenis konten file ketika Anda meng-upload mereka ke Amazon S3 dengan menggunakan parameter `-metadata`, seperti ini:

   ```
   aws s3 cp test.nt s3://bucket-name/my-plain-text-input/test.nt --metadata Content-
   Type=text/plain
   ```

```
aws s3 cp test.rdf s3://bucket-name/my-rdf-input/test.rdf --metadata Content-
Type=application/rdf+xml
```

2. Konfirmasikan bahwa informasi jenis media benar-benar ada. Jalankan:

   ```
   curl -v bucket-name/folder-name
   ```

   Output dari perintah ini harus menunjukkan informasi jenis media yang Anda tetapkan saat mengunggah file.

3. Kemudian Anda dapat menggunakan perintah `SPARQL UPDATE LOAD` untuk mengimpor file-file ini ke Neptune:

   ```
   curl https://your-neptune-endpoint:port/sparql \
     -d "update=LOAD <https://s3.amazonaws.com/bucket-name/my-rdf-input/test.rdf>"
   ```

Langkah-langkah di atas bekerja hanya untuk bucket Amazon S3 publik, atau untuk bucket yang Anda akses menggunakan URL Amazon S3 pre-signed dalam kueri LOAD.

Anda juga dapat mengatur server proksi web untuk memuat dari bucket Amazon S3 privat, seperti yang ditunjukkan di bawah ini:

Menggunakan server web untuk memuat file ke Neptune dengan SPARQL UPDATE LOAD

1. Instal server web pada mesin yang berjalan di dalam VPC yang menjadi host Neptune dan file yang akan dimuat. Misalnya, menggunakan Amazon Linux, Anda dapat menginstal Apache seperti berikut:

   ```
   sudo yum install httpd mod_ssl
   sudo /usr/sbin/apachectl start
   ```

2. Tentukan jenis MIME dari konten file RDF yang akan Anda muat. SPARQL menggunakan header `Content-type` yang dikirim oleh server web untuk menentukan format input konten, sehingga Anda harus menentukan jenis MIME yang relevan untuk Server web.

   Misalnya, misalkan Anda menggunakan ekstensi file berikut untuk mengidentifikasi format file:

   - `.nt`untuk NTriples.
   - `.nq`untuk NQuads.
   - `.trig` untuk Trig.

- `.rdf` untuk RDF/XML.
- `.ttl` untuk Turtle.
- `.n3` untuk N3.
- `.jsonld` untuk JSON-LD.

Jika Anda menggunakan Apache 2 sebagai server web, Anda akan mengedit `/etc/mime.types` file dan menambahkan jenis berikut:

```
text/plain nt
application/n-quads nq
application/trig trig
application/rdf+xml rdf
application/x-turtle ttl
text/rdf+n3 n3
application/ld+json jsonld
```

3. Konfirmasikan bahwa pemetaan tipe MIME bekerja. Setelah server web Anda aktif dan berjalan dan meng-hosting file RDF dalam format pilihan Anda, Anda dapat menguji konfigurasi dengan mengirimkan permintaan ke server web dari host lokal Anda.

   Misalnya, Anda dapat mengirim permintaan seperti ini:

   ```
   curl -v http://localhost:80/test.rdf
   ```

   Kemudian, dalam output detail dari `curl`, Anda akan melihat baris seperti:

   ```
   Content-Type: application/rdf+xml
   ```

   Hal ini menunjukkan bahwa pemetaan jenis konten dinyatakan berhasil.

4. Anda sekarang siap memuat data menggunakan perintah SPARQL UPDATE:

   ```
   curl https://your-neptune-endpoint:port/sparql \
       -d "update=LOAD <http://web_server_private_ip:80/test.rdf>"
   ```

> **Note**
>
> Menggunakan `SPARQL UPDATE LOAD` dapat memicu waktu henti pada server web ketika file sumber yang dimuat berukuran besar. Neptune memproses data file seperti data dialirkan masuk, dan untuk file besar yang dapat memakan waktu lebih lama dari batas waktu yang dikonfigurasi di server. Hal ini pada gilirannya dapat menyebabkan server untuk menutup koneksi, yang dapat mengakibatkan pesan kesalahan berikut ketika Neptune menemui EOF tak terduga di aliran:
>
> ```
> {
>   "detailedMessage":"Invalid syntax in the specified file",
>   "code":"InvalidParameterException"
> }
> ```
>
> Jika Anda menerima pesan ini dan tidak percaya bahwa file sumber berisi sintaks yang tidak valid, coba tingkatkan pengaturan batas waktu di server web. Anda juga dapat mendiagnosis masalah dengan mengaktifkan debug log di server dan mencari waktu henti.

Menggunakan SPARQL UPDATE UNLOAD untuk menghapus data dari Neptune

Neptune juga menyediakan operasi SPARQL kustom, `UNLOAD`, untuk menghapus data yang ditentukan dalam sumber remote. `UNLOAD` dapat dianggap sebagai mitra untuk operasi `LOAD`. Sintaksnya adalah:

```
UNLOAD SILENT (URL of the remote data to be unloaded) FROM GRAPH (named graph from
 which to remove the data)
```

- **`SILENT`** – (Opsional) Menyebabkan operasi mengembalikan sukses bahkan jika ada kesalahan selama pemrosesan data.

  Hal ini dapat berguna ketika transaksi tunggal berisi beberapa pernyataan seperti `"LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;"` dan Anda ingin transaksinya selesai bahkan jika beberapa data jarak jauh tidak dapat diproses.
- *`URL of the remote data to be unloaded`*— (Wajib) Menentukan file data jarak jauh yang berisi data yang akan diturunkan dari grafik.

File jarak jauh harus memiliki salah satu ekstensi berikut (ini adalah format yang sama yang didukung UPDATE-LOAD):

- `.nt`untuk NTriples.
- `.nq`untuk NQuads.
- `.trig` untuk Trig.
- `.rdf` untuk RDF/XML.
- `.ttl` untuk Turtle.
- `.n3` untuk N3.
- `.jsonld` untuk JSON-LD.

Semua data yang berisi file ini akan dihapus dari klaster DB Anda oleh operasi `UNLOAD`.

Otentikasi Amazon S3 harus disertakan dalam URL untuk data yang akan batal dimuat. Anda dapat melakukan pre-sign pada file Amazon S3 lalu menggunakan URL yang dihasilkan untuk mengaksesnya dengan aman. Contoh:

```
aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
 data to unload)
```

Kemudian:

```
curl https://(a Neptune endpoint URL):8182/sparql \
  --data-urlencode 'update=unload (pre-signed URL of the remote Amazon S3 data to be
 unloaded) \
                           from graph (named graph)'
```

Untuk informasi lebih lanjut, lihat Melakukan Autentikasi Permintaan: Menggunakan Parameter Kueri.

- **`FROM GRAPH`** *`(named graph from which to remove the data)`*— (Opsional) Menentukan grafik bernama dari mana data jarak jauh harus diturunkan.

  Neptune mengasosiasikan setiap tripel dengan grafik bernama. Anda dapat menentukan grafik bernama default menggunakan URI grafik bernama fallback, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`, seperti ini:

  ```
  FROM GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
  ```

Dengan cara yang sama saat LOAD sesuai dengan INSERT DATA { *(inline data)* }, UNLOAD sesuai dengan DELETE DATA { *(inline data)* }. Seperti halnya DELETE DATA, UNLOAD tidak berfungsi pada data yang berisi simpul kosong.

Sebagai contoh, jika server web lokal melayani file bernama `data.nt` yang berisi 2 triple berikut:

```
<http://example.org/resource#a> <http://example.org/resource#p> <http://example.org/
resource#b> .
<http://example.org/resource#a> <http://example.org/resource#p> <http://example.org/
resource#c> .
```

Perintah UNLOAD berikut akan menghapus dua triple tersebut dari grafik bernama, `<http://example.org/graph1>`:

```
UNLOAD <http://localhost:80/data.nt> FROM GRAPH <http://example.org/graph1>
```

Hal ini akan memiliki efek yang sama seperti menggunakan perintah DELETE DATA berikut:

```
DELETE DATA {
  GRAPH <http://example.org/graph1> {
    <http://example.org/resource#a> <http://example.org/resource#p> <http://
example.org/resource#b> .
    <http://example.org/resource#a> <http://example.org/resource#p> <http://
example.org/resource#c> .
  }
}
```

Pengecualian yang dilemparkan oleh perintah **UNLOAD**

- **InvalidParameterException** – Ada simpul kosong dalam data. Status HTTP: 400 Permintaan Buruk.

  Pesan: `Blank nodes are not allowed for UNLOAD`

- **InvalidParameterException** – Ada sintaks yang rusak dalam data. Status HTTP: 400 Permintaan Buruk.

  Pesan: `Invalid syntax in the specified file.`

- **`UnloadUrlAccessDeniedException`** – Akses ditolak. Status HTTP: 400 Permintaan Buruk.

  Pesan: `Update failure: Endpoint (Neptune endpoint) reported access denied error. Please verify access.`

- **`BadRequestException`** – Data jarak jauh tidak dapat diambil. Status HTTP: 400 Permintaan Buruk.

  Pesan: (tergantung respons HTTP).

Petunjuk kueri SPARQL

Gunakan petunjuk kueri untuk menentukan optimasi dan evaluasi strategi untuk kueri SPARQL tertentu di Amazon Neptune.

Petunjuk kueri dinyatakan menggunakan pola triple tambahan yang tertanam dalam kueri SPARQL dengan bagian-bagian berikut:

```
scope hint value
```

- lingkup – Menentukan bagian dari kueri yang petunjuk kuerinya diberlakukan padanya, seperti grup tertentu dalam kueri atau kueri lengkap.
- petunjuk – Mengidentifikasi jenis petunjuk yang akan diterapkan.
- nilai – Menentukan perilaku aspek sistem yang sedang dipertimbangkan.

Petunjuk dan cakupan kueri dipaparkan sebagai istilah yang telah ditetapkan dalam namespace Amazon Neptune `http://aws.amazon.com/neptune/vocab/v01/QueryHints#`. Contoh dalam bagian ini menyertakan namespace sebagai awalan `hint` yang didefinisikan dan disertakan dalam kueri:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

Sebagai contoh, hal berikut ini menunjukkan bagaimana cara memasukkan petunjuk `joinOrder` dalam kueri `SELECT`:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

```
SELECT ... {
 hint:Query hint:joinOrder "Ordered" .
 ...
}
```

Kueri sebelumnya menginstruksikan mesin Neptune untuk mengevaluasi gabungan dalam kueri dalam urutan yang diberikan dan menonaktifkan pengurutan ulang otomatis apa pun.

Saat menggunakan petunjuk kueri, pertimbangkan hal berikut:

- Anda dapat menggabungkan petunjuk kueri yang berbeda dalam satu kueri. Misalnya, Anda dapat menggunakan petunjuk kueri `bottomUp` untuk menganotasi subkueri untuk evaluasi bottom-up dan petunjuk kueri `joinOrder` untuk memperbaiki urutan gabungan dalam subkueri.
- Anda dapat menggunakan petunjuk kueri yang sama beberapa kali, dalam cakupan yang tidak tumpang tindih yang berbeda.
- Petunjuk kueri adalah petunjuk. Meskipun mesin kueri umumnya bertujuan untuk mempertimbangkan petunjuk kueri yang diberikan, mesin mungkin juga mengabaikannya.
- Petunjuk kueri adalah pelestarian semantik. Menambahkan petunjuk kueri tidak mengubah output kueri (kecuali untuk urutan hasil potensial ketika tidak ada jaminan pengurutan yang diberikan—yaitu, ketika urutan hasil tidak secara eksplisit ditegakkan menggunakan ORDER BY).

Bagian berikut menyediakan informasi selengkapnya tentang petunjuk kueri yang tersedia dan penggunaannya di Neptune.

Topik

- Cakupan petunjuk kueri SPARQL di Neptune
- Petunjuk kueri SPARQL joinOrder
- Petunjuk kueri SPARQL evaluationStrategy
- Petunjuk kueri SPARQL queryTimeout
- Petunjuk kueri SPARQL rangeSafe
- Petunjuk Kueri SPARQL queryId
- Petunjuk kueri SPARQL useDFE
- Petunjuk kueri SPARQL digunakan dengan DESCRIBE

Cakupan petunjuk kueri SPARQL di Neptune

Tabel berikut menampilkan cakupan yang tersedia, petunjuk terkait, dan deskripsi untuk petunjuk kueri SPARQL di Amazon Neptune. Awalan `hint` dalam entri ini mewakili namespace Neptune untuk petunjuk:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

Lingkup	Petunjuk yang Didukung	Deskripsi
`hint:Query`	JoinOrder	Petunjuk kueri berlaku untuk seluruh kueri.
`hint:Query`	queryTimeout	Nilai batas waktu berlaku untuk seluruh kueri.
`hint:Query`	rangeSafe	Promosi jenis dinonaktifkan untuk seluruh kueri.
`hint:Query`	queryId	Nilai ID kueri berlaku untuk seluruh kueri.
`hint:Query`	useDFE	Penggunaan DFE diaktifka n (atau dinonaktifkan) untuk seluruh kueri.
`hint:Group`	JoinOrder	Permintaan kueri berlaku untuk elemen tingkat atas dalam grup tertentu, tetapi tidak untuk elemen nested (seperti subkueri) atau elemen induk.
`hint:SubQuery`	evaluationStrategy	Petunjuk ditentukan dan diterapkan untuk subkueri SELECT nested. Subkueri dievaluasi secara independe n, tanpa mempertimbangkan

Lingkup	Petunjuk yang Didukung	Deskripsi
		solusi dihitung sebelum subkueri.

Petunjuk kueri SPARQL `joinOrder`

Ketika Anda mengirimkan kueri SPARQL, mesin kueri Amazon Neptune menyelidiki struktur kueri. Mesin tersebut mengurutkan ulang bagian dari kueri dan mencoba untuk meminimalkan jumlah pekerjaan yang diperlukan untuk evaluasi dan waktu respon kueri.

Misalnya, urutan pola tripel yang terhubung biasanya tidak dievaluasi dalam urutan yang diberikan. Urutan tersebut diurutkan ulang menggunakan heuristik dan statistik seperti selektivitas masing-masing pola dan bagaimana mereka terhubung melalui variabel bersama. Selain itu, jika kueri Anda berisi pola yang lebih kompleks seperti subkueri, FILTERs, atau blok OPSIONAL atau MINUS yang kompleks, mesin kueri Neptunus menyusun ulang jika memungkinkan, bertujuan untuk urutan evaluasi yang efisien.

Untuk kueri yang lebih kompleks, urutan di mana Neptune memilih untuk mengevaluasi kueri mungkin tidak selalu optimal. Misalnya, Neptune mungkin kehilangan karakteristik spesifik data instans (seperti mencapai node daya dalam grafik) yang muncul selama evaluasi kueri.

Jika Anda tahu karakteristik data yang tepat dan ingin secara manual mendikte urutan eksekusi kueri, gunakan petunjuk kueri `joinOrder` Neptune untuk menentukan bahwa kueri dievaluasi dalam urutan yang diberikan.

Sintaks petunjuk SPARQ `joinOrder`

Petunjuk kueri `joinOrder` ditentukan sebagai pola tripel yang disetakan dalam kueri SPARQL.

Untuk kejelasan, sintaks berikut menggunakan awalan `hint` yang didefinisikan dan disertakan dalam kueri untuk menentukan namespace petunjuk kueri Neptune:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
scope hint:joinOrder "Ordered" .
```

Cakupan yang Tersedia

- `hint:Query`
- `hint:Group`

Untuk informasi lebih lanjut tentang cakupan petunjuk kueri, lihat Cakupan petunjuk kueri SPARQL di Neptune.

Contoh petunjuk SPARQL `joinOrder`

Bagian ini menunjukkan kueri yang ditulis dengan dan tanpa petunjuk kueri `joinOrder` dan optimasi terkait.

Sebagai contoh, asumsikan bahwa set data berisi hal berikut ini:

- Satu orang bernama `John` yang `:likes` 1.000 orang, termasuk `Jane`.
- Satu orang bernama `Jane` yang `:likes` 10 orang, termasuk `John`.

Tidak Ada Petunjuk Kueri

Kueri SPARQL berikut mengekstrak semua pasangan orang bernama `John` dan `Jane` yang saling menyukai satu sama lain dari satu set data jejaring sosial:

```
PREFIX : <https://example.com/>
SELECT ?john ?jane {
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Mesin kueri Neptune mungkin mengevaluasi pernyataan dalam urutan yang berbeda dari yang ditulis. Sebagai contoh, mesin mungkin memilih untuk menilai dalam susunan berikut:

1. Temukan semua orang yang bernama `John`.
2. Temukan semua orang yang terhubung ke `John` berdasarkan edge `:likes`.
3. Saring set ini berdasarkan orang yang bernama `Jane`.
4. Saring set ini berdasarkan orang yang terhubung ke `John` berdasarkan edge `:likes`.

Menurut set data, mengevaluasi dalam urutan ini menghasilkan 1.000 entitas yang diekstraksi pada langkah kedua. Langkah ketiga mempersempit ini ke node tunggal, `Jane`. Langkah terakhir kemudian menentukan bahwa `Jane` juga `:likes` node `John`.

Petunjuk Kueri

Akan menguntungkan untuk memulai dengan node `Jane` karena dia hanya memiliki 10 edge `:likes` keluar. Hal ini mengurangi jumlah pekerjaan selama evaluasi kueri dengan menghindari ekstraksi 1.000 entitas selama langkah kedua.

Contoh berikut menggunakan petunjuk kueri JoinOrder untuk memastikan bahwa node `Jane` dan edge keluarnya diproses terlebih dulu dengan menonaktifkan semua pengurutan ulang bergabung otomatis untuk kueri:

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?john ?jane {
  hint:Query hint:joinOrder "Ordered" .
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Skenario dunia nyata yang berlaku mungkin berupa aplikasi jaringan sosial di mana orang-orang dalam jaringan diklasifikasikan sebagai influencer dengan banyak koneksi atau sebagai pengguna normal dengan beberapa koneksi. Dalam skenario seperti itu, Anda dapat memastikan bahwa pengguna normal (`Jane`) diproses sebelum influencer (`John`) dalam kueri seperti contoh sebelumnya.

Petunjuk Kueri dan Pengurutan Ulang

Anda dapat mengolah contoh ini satu langkah lebih jauh. Jika Anda tahu bahwa atribut `:name` unik untuk node tunggal, Anda bisa mempercepat kueri dengan pengurutan ulang dan menggunakan petunjuk kueri `joinOrder`. Langkah ini memastikan bahwa node unik diekstraksi pertama.

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?john ?jane {
  hint:Query hint:joinOrder "Ordered" .
  ?person1 :name "Jane" .
  ?person2 :name "John" .
  ?person1 :likes ?person2 .
  ?person2 :likes ?person1 .
}
```

Dalam kasus ini, Anda dapat mengurangi kueri untuk tindakan tunggal berikut di setiap langkah:

1. Cari node satu orang dengan `:name Jane`.
2. Cari node satu orang dengan `:name John`.
3. Pastikan bahwa node pertama terhubung ke node kedua dengan edge `:likes`.
4. Pastikan bahwa node kedua terhubung ke node pertama dengan edge `:likes`.

> **Important**
>
> Jika Anda memilih urutan yang salah, petunjuk kueri `joinOrder` dapat menyebabkan penurunan kinerja yang signifikan. Sebagai contoh, contoh sebelumnya akan menjadi tidak efisien jika atribut `:name` tidak unik. Jika semua 100 node diberi nama `Jane` dan semua 1.000 node diberi nama `John`, maka akhir kueri akan memeriksa 1.000 * 100 (100.000) pasang untuk edge `:likes`.

Petunjuk kueri SPARQL **`evaluationStrategy`**

Petunjuk kueri `evaluationStrategy` untuk memberi tahu mesin Amazon Neptune bahwa fragmen kueri yang dianotasi harus dievaluasi dari bawah ke atas sebagai unit independen. Ini berarti bahwa tidak ada solusi dari langkah-langkah evaluasi sebelumnya yang digunakan untuk menghitung fragmen kueri. Fragmen kueri dievaluasi sebagai unit mandiri, dan solusi yang dihasilkan digabungkan dengan sisa kueri setelah dihitung.

Menggunakan petunjuk kueri `evaluationStrategy` menyiratkan rencana kueri (non-pipelined) pemblokiran, yang berarti bahwa solusi dari fragmen yang dianotasi dengan petunjuk kueri dimeterialisasikan dan di-buffer dalam memori utama. Menggunakan petunjuk kueri ini mungkin secara signifikan meningkatkan jumlah memori utama yang diperlukan untuk mengevaluasi kueri, terutama jika fragmen kueri yang dianotasi menghitung sejumlah besar hasil.

Sintaks petunjuk SPARQ **`evaluationStrategy`**

Petunjuk kueri `evaluationStrategy` ditentukan sebagai pola tripel yang disetakan dalam kueri SPARQL.

Untuk kejelasan, sintaks berikut menggunakan awalan `hint` yang didefinisikan dan disertakan dalam kueri untuk menentukan namespace petunjuk kueri Neptune:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

```
hint:SubQuery hint:evaluationStrategy "BottomUp" .
```

Cakupan yang Tersedia

- `hint:SubQuery`

> **Note**
>
> Petunjuk kueri ini didukung hanya dalam subqueries nested.

Untuk informasi lebih lanjut tentang cakupan petunjuk kueri, lihat Cakupan petunjuk kueri SPARQL di Neptune.

Contoh petunjuk SPARQL **`evaluationStrategy`**

Bagian ini menunjukkan kueri yang ditulis dengan dan tanpa petunjuk kueri `evaluationStrategy` dan optimasi terkait.

Untuk contoh ini, anggaplah set data memiliki karakteristik sebagai berikut:

- Berisi 1.000 edge berlabel `:connectedTo`.
- Setiap node `component` terhubung ke rata-rata 100 node `component` lainnya.
- Jumlah umum koneksi siklus empat hop antara node adalah sekitar 100.

Sebagai contoh umum, petunjuk `evaluationStrategy` dapat membantu mengoptimalkan pola kueri yang mengandung siklus.

Tidak Ada Petunjuk Kueri

Kueri SPARQL berikut mengekstrak semua `component` node yang terhubung secara siklus satu sama lain melalui empat hop:

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
  ?component1 :connectedTo ?component2 .
  ?component2 :connectedTo ?component3 .
```

```
  ?component3 :connectedTo ?component4 .
  ?component4 :connectedTo ?component1 .
}
```

Pendekatan mesin kueri Neptune adalah mengevaluasi kueri ini menggunakan langkah-langkah berikut:

- Ekstrak kesemua 1.000 edge `connectedTo` dalam grafik.
- Perluas dengan 100x (jumlah edge `connectedTo` keluar dari component2).

  Hasil antara: 100.000 node.
- Perluas dengan 100x (jumlah edge `connectedTo` keluar dari component3).

  Hasil antara: 10.000.000 node.
- Pindai 10.000.000 node untuk siklus menutup.

Hal ini menyebabkan rencana kueri streaming, yang memiliki jumlah konstan memori utama.

Petunjuk Kueri dan Subqueries

Anda mungkin ingin menukar ruang memori utama untuk komputasi yang lebih cepat. Dengan menulis ulang kueri menggunakan petunjuk kueri `evaluationStrategy`, Anda dapat memaksa mesin untuk menghitung gabungan antara dua subset kecil yang dimaterialisasi.

```
PREFIX : <https://example.com/>
          PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
  {
    SELECT * WHERE {
      hint:SubQuery hint:evaluationStrategy "BottomUp" .
      ?component1 :connectedTo ?component2 .
      ?component2 :connectedTo ?component3 .
    }
  }
  {
    SELECT * WHERE {
      hint:SubQuery hint:evaluationStrategy "BottomUp" .
      ?component3 :connectedTo ?component4 .
      ?component4 :connectedTo ?component1 .
    }
  }
```

```
}
```

Alih-alih mengevaluasi pola triple secara berurutan sembari secara iteratif menggunakan hasil dari pola triple sebelumnya sebagai input untuk pola yang akan datang, petunjuk `evaluationStrategy` menyebabkan dua subqueries dievaluasi secara independen. Kedua subqueries menghasilkan 100.000 node untuk hasil antara, yang kemudian digabungkan bersama-sama untuk membentuk output akhir.

Secara khusus, ketika Anda menjalankan Neptune pada tipe instans yang lebih besar, menyimpan sementara kedua 100.000 subset ini dalam memori utama meningkatkan penggunaan memori sebagai akibat dari secara signifikan mempercepat evaluasi.

Petunjuk kueri SPARQL **`queryTimeout`**

Petunjuk kueri `queryTimeout` menentukan batas waktu yang lebih pendek dari nilai `neptune_query_timeout` yang ditetapkan dalam grup parameter DB.

Jika kueri diakhiri sebagai hasil dari petunjuk ini, `TimeLimitExceededException` dilemparkan, dengan pesan `Operation terminated (deadline exceeded)`.

Sintaks petunjuk SPARQ **`queryTimeout`**

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ... WHERE {
    hint:Query hint:queryTimeout 10 .
    # OR
    hint:Query hint:queryTimeout "10" .
    # OR
    hint:Query hint:queryTimeout "10"^^xsd:integer .
 ...
}
```

Nilai batas waktu dinyatakan dalam milidetik.

Nilai batas waktu harus lebih kecil dari nilai `neptune_query_timeout` yang ditetapkan dalam grup parameter DB. Jika tidak, pengecualian `MalformedQueryException` dilemparkan dengan pesan `Malformed query: Query hint 'queryTimeout' must be less than neptune_query_timeout DB Parameter Group`.

Petunjuk kueri `queryTimeout` harus ditentukan dalam klausa `WHERE` dari kueri utama, atau dalam klausa `WHERE` dari salah satu subqueries seperti yang ditunjukkan pada contoh di bawah ini.

Ini harus diatur hanya sekali di semua bagian Pembaruan queries/subqueries dan SPARQL (seperti INSERT dan DELETE). Jika tidak, pengecualian `MalformedQueryException` dilemparkan dengan pesan `Malformed query: Query hint 'queryTimeout' must be set only once`.

Cakupan yang Tersedia

Petunjuk `queryTimeout` dapat diterapkan baik untuk kueri dan update SPARQL.

- Dalam kueri SPARQL, petunjuk dapat muncul di klausa WHERE dari kueri utama atau subkueri.
- Dalam update SPARQL, petunjuk dapat diatur dalam klausa INSERT, DELETE, atau WHERE. Jika ada beberapa klausa pembaruan, petunjuk hanya dapat diatur dalam salah satu dari klausa tersebut.

Untuk informasi lebih lanjut tentang cakupan petunjuk kueri, lihat Cakupan petunjuk kueri SPARQL di Neptune.

Contoh petunjuk SPARQL **`queryTimeout`**

Berikut adalah contoh penggunaan `hint:queryTimeout` di klausa `WHERE` utama dari kueri `UPDATE`:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
INSERT {
    ?s ?p ?o
} WHERE {
    hint:Query hint:queryTimeout 100 .
    ?s ?p ?o .
}
```

Di sini, `hint:queryTimeout` berada di klausa `WHERE` dari subkueri:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
   ?s ?p ?o .
   {
      SELECT ?s WHERE {
         hint:Query hint:queryTimeout 100 .
         ?s ?p1 ?o1 .
      }
```

```
    }
}
```

Petunjuk kueri SPARQL **rangeSafe**

Gunakan petunjuk kueri ini untuk menonaktifkan promosi jenis untuk kueri SPARQL.

Ketika Anda mengirimkan kueri SPARQL yang mencakup `FILTER` di atas nilai atau jangkauan numerik, mesin kueri Neptune biasanya harus menggunakan promosi jenis ketika mengeksekusi kueri. Ini berarti bahwa mesin harus memeriksa nilai-nilai dari setiap jenis yang dapat memegang nilai yang Anda gunakan untuk menyaring.

Misalnya, jika Anda menyaring untuk nilai yang sama dengan 55, mesin harus mencari bilangan bulat yang sama dengan 55, bilangan bulat panjang sama dengan 55L, mengapung sama dengan 55.0, dan sebagainya. Setiap jenis promosi memerlukan pencarian tambahan pada penyimpanan, yang dapat menyebabkan permintaan yang tampaknya sederhana tiba-tiba mengambil waktu yang lama untuk menyelesaikan.

Sering kali promosi jenis tidak perlu karena Anda tahu sebelumnya bahwa Anda hanya perlu menemukan nilai dari satu jenis tertentu. Ketika hal ini terjadi, Anda dapat mempercepat kueri Anda secara dramatis dengan menggunakan petunjuk kueri `rangeSafe` untuk mematikan promosi jenis.

Sintaks petunjuk SPARQ **rangeSafe**

Petunjuk kueri `rangeSafe` mengambil nilai `true` untuk menonaktifkan promosi jenis. Hal ini juga menerima nilai `false` (default).

Contoh. Contoh berikut menunjukkan bagaimana untuk mematikan promosi jenis ketika penyaringan untuk nilai integer dari o lebih besar dari 1:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
   ?s ?p ?o .
   hint:Prior hint:rangeSafe 'true' .
   FILTER (?o > '1'^^<http://www.w3.org/2001/XMLSchema#int>)
```

Petunjuk Kueri SPARQL **queryId**

Gunakan petunjuk kueri ini untuk menetapkan nilai querild Anda sendiri ke kueri SPARQL.

Contoh:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * WHERE {
  hint:Query hint:queryId "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
  {?s ?p ?o}}
```

Nilai yang Anda tetapkan harus unik di semua kueri di Neptune DB.

Petunjuk kueri SPARQL **`useDFE`**

Gunakan petunjuk kueri ini untuk mengaktifkan penggunaan DFE untuk mengeksekusi kueri. Secara default Neptunus tidak menggunakan DFE tanpa petunjuk kueri ini disetel ke, karena parameter instance neptune_dfe_query_engine default `true` ke. `viaQueryHint` Jika Anda menyetel parameter instance itu`enabled`, mesin DFE digunakan untuk semua kueri kecuali yang memiliki petunjuk `useDFE` kueri yang disetel ke. `false`

Contoh mengaktifkan penggunaan DFE untuk kueri:

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>

SELECT ?john ?jane
{
  hint:Query hint:useDFE true .
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Petunjuk kueri SPARQL digunakan dengan DESCRIBE

`DESCRIBE`Kueri SPARQL menyediakan mekanisme yang fleksibel untuk meminta deskripsi sumber daya. Namun, spesifikasi SPARQL tidak menentukan semantik yang tepat dari. `DESCRIBE`

Dimulai dengan rilis mesin 1.2.0.2, Neptunus mendukung beberapa `DESCRIBE` mode dan algoritma berbeda yang cocok untuk situasi yang berbeda.

Dataset sampel ini dapat membantu mengilustrasikan mode yang berbeda:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <https://example.com/> .

:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JohnDoe :firstName "John" .
:JaneDoe :knows _:b1 .
_:b1 :knows :RichardRoe .

:RichardRoe :knows :JaneDoe .
:RichardRoe :firstName "Richard" .

_:s1 rdf:type rdf:Statement .
_:s1 rdf:subject :JaneDoe .
_:s1 rdf:predicate :knows .
_:s1 rdf:object :JohnDoe .
_:s1 :knowsFrom "Berlin" .

:ref_s2 rdf:type rdf:Statement .
:ref_s2 rdf:subject :JaneDoe .
:ref_s2 rdf:predicate :knows .
:ref_s2 rdf:object :JohnDoe .
:ref_s2 :knowsSince 1988 .
```

Contoh di bawah ini mengasumsikan bahwa deskripsi sumber daya `:JaneDoe` diminta menggunakan kueri SPARQL seperti ini:

```
DESCRIBE <https://example.com/JaneDoe>
```

Petunjuk kueri SPARQL **`describeMode`**

Petunjuk kueri `hint:describeMode` SPARQL digunakan untuk memilih salah satu mode `DESCRIBE` SPARQL berikut yang didukung oleh Neptunus:

Mode **`ForwardOneStep`** DESCRIPTE

Anda menjalankan `ForwardOneStep` mode dengan petunjuk `describeMode` kueri seperti ini:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "ForwardOneStep"
```

```
}
```

`ForwardOneStep`Mode hanya mengembalikan atribut dan tautan penerusan sumber daya yang akan dijelaskan. Dalam kasus contoh, ini berarti mengembalikan tiga kali lipat yang memiliki`:JaneDoe`, sumber daya yang akan dijelaskan, sebagai subjek:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b301990159 .
```

Perhatikan bahwa kueri DESCRIBE dapat mengembalikan tiga kali lipat dengan node kosong, seperti`_:b301990159`, yang berbeda IDs setiap kali, dibandingkan dengan kumpulan data input.

Mode **`SymmetricOneStep`** DESCRIPTE

`SymmetricOneStep`adalah mode DESCRIBE default jika Anda tidak memberikan petunjuk kueri. Anda juga dapat memanggilnya secara eksplisit dengan petunjuk `describeMode` kueri seperti ini:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SymmetricOneStep"
}
```

Di bawah `SymmetricOneStep` semantik, `DESCRIBE` mengembalikan atribut, tautan maju, dan tautan balik sumber daya yang akan dijelaskan:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b318767375 .

_:b318767631 rdf:subject :JaneDoe .

:RichardRoe :knows :JaneDoe .

:ref_s2 rdf:subject :JaneDoe .
```

Deskripsi Terbatas Ringkas (**`CBD`**) mode DESCRIPTION

Mode Concise Bounded Description (`CBD`) dipanggil menggunakan petunjuk `describeMode` kueri seperti ini:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "CBD"
}
```

Di bawah `CBD` semantik, `DESCRIBE` mengembalikan Deskripsi Terbatas Ringkas (seperti yang didefinisikan oleh W3C) dari sumber daya yang akan dijelaskan:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b285212943 .
_:b285212943 :knows :RichardRoe .

_:b285213199 rdf:subject :JaneDoe .
_:b285213199 rdf:type rdf:Statement .
_:b285213199 rdf:predicate :knows .
_:b285213199 rdf:object :JohnDoe .
_:b285213199 :knowsFrom "Berlin" .

:ref_s2 rdf:subject :JaneDoe .
```

Deskripsi Terbatas Ringkas dari sumber daya RDF (yaitu, simpul dalam grafik RDF) adalah subgraf terkecil yang berpusat di sekitar node yang dapat berdiri sendiri. Dalam praktiknya ini berarti bahwa jika Anda menganggap grafik ini sebagai pohon, dengan simpul yang ditunjuk sebagai akar, tidak ada simpul kosong (bnode) sebagai daun pohon itu. Karena bnodes tidak dapat ditangani secara eksternal atau digunakan dalam kueri berikutnya, itu tidak cukup untuk menelusuri grafik hanya untuk menemukan hop tunggal berikutnya dari node saat ini. Anda juga harus pergi cukup jauh untuk menemukan sesuatu yang dapat digunakan dalam kueri berikutnya (yaitu, sesuatu selain bnode).

Menghitung CBD

Diberikan node tertentu (node awal atau root) dalam grafik RDF sumber, CBD dari node tersebut dihitung sebagai berikut:

1. Sertakan dalam subgraf semua pernyataan dalam grafik sumber di mana subjek pernyataan adalah simpul awal.
2. Secara rekursif, untuk semua pernyataan dalam subgraf sejauh ini yang memiliki objek simpul kosong, sertakan dalam subgraf semua pernyataan dalam grafik sumber di mana subjek pernyataan adalah simpul kosong, dan yang belum termasuk dalam subgraf.

3. Secara rekursif, untuk semua pernyataan yang termasuk dalam subgraf sejauh ini, untuk semua reifikasi pernyataan ini dalam grafik sumber, sertakan CBD yang dimulai dari simpul setiap reifikasi. `rdf:Statement`

Ini menghasilkan subgraf di mana node objek adalah referensi IRI atau literal, atau node kosong yang tidak berfungsi sebagai subjek dari pernyataan apa pun dalam grafik. Perhatikan bahwa CBD tidak dapat dihitung menggunakan kueri SPARQL SELECT atau CONSTRUCT tunggal.

Deskripsi Terbatas Ringkas Simetris () mode DESCRIBE **`SCBD`**

Mode Symmetric Concise Bounded Description (`SCBD`) dipanggil menggunakan petunjuk `describeMode` kueri seperti ini:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SCBD"
}
```

Di bawah `SCBD` semantik, `DESCRIBE` mengembalikan Deskripsi Terbatas Ringkas Simetris dari sumber daya (seperti yang didefinisikan oleh W3C dalam Menjelaskan Kumpulan Data Tertaut dengan Kosakata VoID:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b335544591 .
_:b335544591 :knows :RichardRoe .

:RichardRoe :knows :JaneDoe .

_:b335544847 rdf:subject :JaneDoe .
_:b335544847 rdf:type rdf:Statement .
_:b335544847 rdf:predicate :knows .
_:b335544847 rdf:object :JohnDoe .
_:b335544847 :knowsFrom "Berlin" .

:ref_s2 rdf:subject :JaneDoe .
```

Keuntungan CBD dan SCBD dibandingkan `SymmetricOneStep` mode `ForwardOneStep` dan adalah bahwa node kosong selalu diperluas untuk memasukkan representasi mereka. Ini mungkin

merupakan keuntungan penting karena Anda tidak dapat menanyakan simpul kosong menggunakan SPARQL. Selain itu, mode CBD dan SCBD juga mempertimbangkan reifikasi.

Perhatikan bahwa petunjuk `describeMode` kueri juga dapat menjadi bagian dari `WHERE` klausa:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE ?s
WHERE {
  hint:Query hint:describeMode "CBD" .
  ?s rdf:type <https://example.com/Person>
}
```

Petunjuk kueri SPARQL **`describeIterationLimit`**

Petunjuk kueri `hint:describeIterationLimit` SPARQL memberikan batasan opsional pada jumlah maksimum ekspansi berulang yang akan dilakukan untuk algoritme DESCRIBE berulang seperti CBD dan SCBD.

Batas DESKRIPSIKAN adalah ANDed bersama-sama. Oleh karena itu, jika batas iterasi dan batas pernyataan ditentukan, maka kedua batas harus dipenuhi sebelum kueri DESCRIBE terputus.

Default untuk nilai ini adalah 5. Anda dapat mengaturnya ke NOL (0) untuk menentukan NO limit pada jumlah ekspansi iteratif.

Petunjuk kueri SPARQL **`describeStatementLimit`**

Petunjuk kueri `hint:describeStatementLimit` SPARQL memberikan kendala opsional pada jumlah maksimum pernyataan yang mungkin ada dalam respons kueri DESCRIBE. Ini hanya diterapkan untuk algoritma DESCRIBE berulang seperti CBD dan SCBD.

Batas DESKRIPSIKAN adalah ANDed bersama-sama. Oleh karena itu, jika batas iterasi dan batas pernyataan ditentukan, maka kedua batas harus dipenuhi sebelum kueri DESCRIBE terputus.

Default untuk nilai ini adalah 5000. Anda dapat mengaturnya ke NOL (0) untuk menentukan NO limit pada jumlah pernyataan yang dikembalikan.

Perilaku DESKRIPSI SPARQL sehubungan dengan grafik default

Formulir `DESCRIBE`kueri SPARQL memungkinkan Anda mengambil informasi tentang sumber daya tanpa mengetahui struktur data dan tanpa harus membuat kueri. Bagaimana informasi ini dirakit diserahkan kepada implementasi SPARQL. Neptunus menyediakan beberapa petunjuk kueri yang memanggil mode dan algoritma yang berbeda untuk digunakan. `DESCRIBE`

Dalam implementasi Neptunus, terlepas dari modenya, `DESCRIBE` hanya menggunakan data yang ada dalam grafik default SPARQL. Ini konsisten dengan cara SPARQL memperlakukan kumpulan data (lihat Menentukan Kumpulan Data RDF dalam spesifikasi SPARQL).

Di Neptunus, grafik default berisi semua tripel unik dalam penyatuan semua grafik bernama dalam database, kecuali grafik bernama tertentu ditentukan menggunakan klausa. `FROM` and/or `FROM NAMED` Semua data RDF di Neptunus disimpan dalam grafik bernama. Jika rangkap tiga dimasukkan tanpa konteks grafik bernama, Neptunus menyimpannya dalam grafik bernama yang ditunjuk. `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`

Ketika satu atau beberapa grafik bernama ditentukan menggunakan `FROM` klausa, grafik default adalah gabungan dari semua tiga kali lipat unik dalam grafik bernama tersebut. Jika tidak ada `FROM` klausa dan ada satu atau lebih `FROM NAMED` klausa, maka grafik default kosong.

Contoh **`DESCRIBE`** SPARQL

Pertimbangkan data berikut:

```
PREFIX ex: <https://example.com/>

GRAPH ex:g1 {
    ex:s ex:p1 "a" .
    ex:s ex:p2 "c" .
}

GRAPH ex:g2 {
    ex:s ex:p3 "b" .
    ex:s ex:p2 "c" .
}

ex:s ex:p3 "d" .
```

Untuk kueri ini:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM ex:g1
FROM NAMED ex:g2
WHERE {
  GRAPH ex:g2 { ?s ?p "b" . }
}
```

Neptunus akan kembali:

```
ex:s ex:p1 "a" .
ex:s ex:p2 "c" .
```

Di sini, pola grafik `GRAPH ex:g2 { ?s ?p "b" }` dievaluasi terlebih dahulu, menghasilkan binding untuk`?s`, dan kemudian `DESCRIBE` bagian tersebut dievaluasi melalui grafik default, yang sekarang hanya. `ex:g1`

Namun, untuk kueri ini:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM NAMED ex:g1
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

Neptunus tidak akan mengembalikan apa pun, karena ketika `FROM NAMED` klausa hadir tanpa klausa apa `FROM` pun, grafik default kosong.

Dalam query berikut, `DESCRIBE` digunakan dengan no `FROM` atau `FROM NAMED` klausa hadir:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

Dalam situasi ini, grafik default terdiri dari semua tiga kali lipat unik dalam penyatuan semua grafik bernama dalam database (secara formal, penggabungan RDF), sehingga Neptunus akan kembali:

```
ex:s ex:p1 "a" .
ex:s ex:p2 "c" .
ex:s ex:p3 "b" .
ex:s ex:p3 "d" .
```

API status kueri SPARQL

Untuk mendapatkan status kueri SPARQL, gunakan HTTP `GET` atau `POST` untuk membuat permintaan ke titik akhir `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql/status`.

Parameter permintaan status kueri SPARQL

queryId (opsional)

ID dari kueri SPARQL yang berjalan. Hanya menampilkan status kueri yang ditentukan.

Sintaks respons status kueri SPARQL

```
{
    "acceptedQueryCount": integer,
    "runningQueryCount": integer,
    "queries": [
      {
        "queryId":"guid",
        "queryEvalStats":
          {
            "subqueries": integer,
            "elapsed": integer,
            "cancelled": boolean
          },
        "queryString": "string"
      }
    ]
}
```

Nilai respons status kueri SPARQL

acceptedQueryCount

Jumlah kueri yang diterima sejak restart terakhir mesin Neptune.

runningQueryCount

Jumlah kueri SPARQL yang sedang berjalan.

pertanyaan

Daftar kueri SPARQL saat ini.

queryId

id GUID untuk kueri. Neptune secara otomatis memberikan nilai ID ini ke setiap kueri, atau Anda juga dapat menetapkan ID Anda sendiri (lihat Menyuntikkan ID Kustom Ke Dalam Gremlin Neptune atau Kueri SPARQL).

queryEvalStats

Statistik untuk kueri ini.

subkueri

Jumlah subkueri dalam kueri ini.

berlalu

Jumlah milidetik kueri telah berjalan sejauh ini.

dibatalkan

True menunjukkan bahwa kueri dibatalkan.

queryString

Kueri yang diajukan.

Contoh status kueri SPARQL

Berikut ini adalah contoh perintah status menggunakan `curl` dan `GET` HTTP.

```
curl https://your-neptune-endpoint:port/sparql/status
```

Output ini menunjukkan satu kueri yang berjalan.

```
{
    "acceptedQueryCount":9,
    "runningQueryCount":1,
    "queries": [
        {
            "queryId":"fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
            "queryEvalStats":
                {
                    "subqueries": 0,
                    "elapsed": 29256,
                    "cancelled": false
                },
            "queryString": "SELECT ?s ?p ?o WHERE {?s ?p ?o}"
        }
    ]
}
```

Pembatalan kueri SPARQL

Untuk mendapatkan status kueri SPARQL, gunakan HTTP `GET` atau `POST` untuk membuat permintaan ke titik akhir `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql/status`.

Parameter permintaan pembatalan kueri SPARQL

CancelQuery

(Diperlukan) Memberi tahu perintah status untuk membatalkan kueri. Parameter ini tidak mengambil nilai.

queryId

(Diperlukan) ID dari kueri SPARQL yang berjalan yang akan dibatalkan.

diam

(Opsional) Jika `silent=true` maka kueri yang berjalan dibatalkan dan kode respon HTTPnya adalah 200. Jika `silent` tidak ada atau `silent=false`, kueri dibatalkan dengan kode status HTTP 500.

Contoh pembatalan kueri SPARQL

Contoh 1: Pembatalan dengan **`silent=false`**

Berikut ini adalah contoh perintah status menggunakan `curl` untuk membatalkan kueri dengan parameter `silent` diatur ke `false`:

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
  -d "silent=false"
```

Kecuali kueri sudah mulai men-streaming hasil, kueri yang dibatalkan kemudian akan mengembalikan kode HTTP 500 dengan respons seperti ini:

```
{
  "code": "CancelledByUserException",
  "requestId": "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47",
  "detailedMessage": "Operation terminated (cancelled by user)"
}
```

Jika kueri sudah mengembalikan kode HTTP 200 (OK) dan telah mulai men-streaming hasil sebelum dibatalkan, informasi pengecualian batas waktu dikirim ke stream output biasa.

Contoh 2: Pembatalan dengan **`silent=true`**

Berikut ini adalah contoh dari perintah status yang sama seperti di atas kecuali dengan parameter `silent` sekarang diatur ke `true`:

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
  -d "silent=true"
```

Perintah ini akan mengembalikan respons yang sama seperti ketika `silent=false`, tetapi kueri yang dibatalkan sekarang akan mengembalikan kode HTTP 200 dengan respons seperti ini:

```
{
  "head" : {
    "vars" : [ "s", "p", "o" ]
  },
  "results" : {
    "bindings" : [ ]
  }
}
```

Menggunakan Protokol HTTP (GSP) SPARQL 1.1 Graph Store di Amazon Neptunus

Dalam rekomendasi Protokol HTTP SPARQL 1.1 Graph Store, W3C mendefinisikan protokol HTTP untuk mengelola grafik RDF. Ini mendefinisikan operasi untuk menghapus, membuat, dan mengganti konten grafik RDF serta untuk menambahkan pernyataan RDF ke konten yang ada.

Protokol graph-store (GSP) menyediakan cara mudah untuk memanipulasi seluruh grafik Anda tanpa harus menulis kueri SPARQL yang kompleks.

Neptunus sepenuhnya mendukung protokol ini.

Titik akhir untuk protokol graph-store (GSP) adalah:

```
https://your-neptune-cluster:port/sparql/gsp/
```

Untuk mengakses grafik default dengan GSP, gunakan:

```
https://your-neptune-cluster:port/sparql/gsp/?default
```

Untuk mengakses grafik bernama dengan GSP, gunakan:

```
https://your-neptune-cluster:port/sparql/gsp/?graph=named-graph-URI
```

Detail khusus implementasi GSP Neptunus

Neptunus sepenuhnya mengimplementasikan rekomendasi W3C yang mendefinisikan GSP. Namun, ada beberapa situasi yang spesifikasi tidak mencakup.

Salah satunya adalah kasus di mana `POST` permintaan `PUT` atau menentukan satu atau beberapa grafik bernama di badan permintaan yang berbeda dari grafik yang ditentukan oleh URL permintaan. Ini hanya dapat terjadi ketika format RDF badan permintaan mendukung grafik bernama, seperti, misalnya, menggunakan atau. `Content-Type: application/n-quads` `Content-Type: application/trig`

Dalam situasi ini, Neptunus menambahkan atau memperbarui semua grafik bernama yang ada di tubuh, serta grafik bernama yang ditentukan dalam URL.

Misalnya, misalkan dimulai dengan database kosong, Anda mengirim `PUT` permintaan untuk meningkatkan suara menjadi tiga grafik. Satu, bernama`urn:votes`, berisi semua suara dari semua tahun pemilihan. Dua lainnya, bernama `urn:votes:2005` dan`urn:votes:2019`, berisi suara dari tahun pemilihan tertentu. Permintaan dan muatannya terlihat seperti ini:

```
PUT "http://your-Neptune-cluster:port/sparql/gsp/?graph=urn:votes"
  Host: example.com
  Content-Type: application/n-quads

  PAYLOAD:

  <urn:JohnDoe> <urn:votedFor> <urn:Labour> <urn:votes:2005>
  <urn:JohnDoe> <urn:votedFor> <urn:Conservative> <urn:votes:2019>
  <urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes:2005>
  <urn:JaneSmith> <urn:votedFor> <urn:Conservative> <urn:votes:2019>
```

Setelah permintaan dijalankan, data dalam database terlihat seperti ini:

```
<urn:JohnDoe>   <urn:votedFor> <urn:Labour>           <urn:votes:2005>
<urn:JohnDoe>   <urn:votedFor> <urn:Conservative>     <urn:votes:2019>
<urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes:2005>
<urn:JaneSmith> <urn:votedFor> <urn:Conservative>     <urn:votes:2019>
<urn:JohnDoe>   <urn:votedFor> <urn:Labour>           <urn:votes>
<urn:JohnDoe>   <urn:votedFor> <urn:Conservative>     <urn:votes>
<urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes>
<urn:JaneSmith> <urn:votedFor> <urn:Conservative>     <urn:votes>
```

Situasi ambigu lainnya adalah di mana lebih dari satu grafik ditentukan dalam URL permintaan itu sendiri, menggunakan salah satu dari`PUT`,`POST`, `GET` atau. `DELETE` Contoh:

```
POST "http://your-Neptune-cluster:port/sparql/gsp/?
graph=urn:votes:2005&graph=urn:votes:2019"
```

Atau:

```
GET "http://your-Neptune-cluster:port/sparql/gsp/?default&graph=urn:votes:2019"
```

Dalam situasi ini, Neptunus mengembalikan HTTP 400 dengan pesan yang menunjukkan bahwa hanya satu grafik yang dapat ditentukan dalam URL permintaan.

Menganalisis eksekusi kueri Neptune menggunakan **`explain`** SPARQL

Amazon Neptune telah menambahkan fitur SPARQL bernama explain. Fitur ini adalah alat swalayan untuk memahami pendekatan eksekusi yang diambil oleh mesin Neptune. Anda meminta dengan menambahkan parameter `explain` ke panggilan HTTP yang mengirimkan kueri SPARQL.

Fitur `explain` menyediakan informasi tentang struktur logis rencana eksekusi kueri. Anda dapat menggunakan informasi ini untuk mengidentifikasi potensi hambatan evaluasi dan eksekusi. Anda kemudian dapat menggunakan petunjuk kueri untuk meningkatkan rencana eksekusi kueri Anda.

Topik

- Memahami bagaimana mesin kueri SPARQL Neptune bekerja.
- Cara menggunakan explain SPARQL untuk menganalisis eksekusi kueri Neptune
- Contoh explain SPARQL yang dipanggil di Neptune
- Operator explain SPARQL Neptune
- Keterbatasan explain SPARQL di Neptune

Memahami bagaimana mesin kueri SPARQL Neptune bekerja.

Untuk menggunakan informasi yang disediakan fitur `explain` SPARQL, Anda perlu memahami beberapa rincian tentang cara mesin kueri Amazon Neptune SPARQL bekerja.

Mesin menerjemahkan setiap kueri SPARQL ke dalam alur operator. Mulai dari operator pertama, solusi perantara yang dikenal sebagai daftar pengikatan mengalir melalui alur operator ini. Anda dapat memikirkan daftar pengikatan sebagai tabel di mana header tabel adalah bagian dari variabel yang digunakan dalam kueri. Setiap baris dalam tabel mewakili hasil, sampai titik evaluasi.

Mari kita asumsikan bahwa dua prefiks namespace telah didefinisikan untuk data kami:

```
@prefix ex:   <http://example.com> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

Berikut ini akan menjadi contoh dari daftar pengikatan sederhana dalam konteks ini:

```
?person       | ?firstName
------------------------------------------------------
ex:JaneDoe    | "Jane"
ex:JohnDoe    | "John"
ex:RichardRoe | "Richard"
```

Untuk masing-masing dari tiga orang, daftar tersebut mengikat variabel `?person` ke pengenal dari orang tersebut, dan variabel `?firstName` ke nama depat orang tersebut.

Dalam kasus umum, variabel dapat tetap tidak terikat, jika, misalnya, ada pemilihan `OPTIONAL` variabel dalam kueri yang tanpa nilai dalam data.

Operator `PipelineJoin` adalah contoh dari operator mesin kueri Neptune yang ada dalam output `explain`. Operator tersebut dibutuhkan sebagai set mengikat masuk dari operator sebelumnya dan bergabung dengan pola triple, misalnya `(?person, foaf:lastName, ?lastName)`. Operasi ini menggunakan pengikatan untuk variabel `?person` dalam stream inputnya, menggantikannya ke dalam pola triple, dan mencari triple dari database.

Ketika dieksekusi dalam konteks pengikatan masuk dari tabel sebelumnya, `PipelineJoin`akan mengevaluasi tiga pencarian, yaitu berikut ini:

```
(ex:JaneDoe,    foaf:lastName, ?lastName)
(ex:JohnDoe,    foaf:lastName, ?lastName)
```

```
  (ex:RichardRoe, foaf:lastName, ?lastName)
```

Pendekatan ini disebut evaluasi as-bound. Solusi dari proses evaluasi ini digabungkan kembali dengan solusi yang masuk, mengalasi `?lastName` yang terdeteksi dalam solusi yang masuk. Dengan asumsi bahwa Anda menemukan nama terakhir untuk ketiga orang tersebut, operator akan menghasilkan daftar mengikat keluar yang akan terlihat seperti ini:

```
?person       | ?firstName | ?lastName
-------------------------------------
ex:JaneDoe    | "Jane"     | "Doe"
ex:JohnDoe    | "John"     | "Doe"
ex:RichardRoe | "Richard"  | "Roe"
```

Daftar mengikat keluar ini kemudian berfungsi sebagai input untuk operator berikutnya dalam alur. Pada akhirnya, output dari operator terakhir dalam alur mendefinisikan hasil kueri.

Alur operator sering linier, dalam arti bahwa setiap operator mengeluarkan solusi untuk satu operator terhubung. Namun, dalam beberapa kasus, mereka dapat memiliki struktur yang lebih kompleks. Misalnya, operator `UNION` dalam kueri SPARQL dipetakan ke operasi `Copy`. Operasi ini menduplikat binding dan meneruskan salinan menjadi dua subplan, satu untuk sisi kiri dan yang lainnya untuk sisi kanan `UNION`.

Untuk informasi lebih lanjut tentang operator, lihat Operator `explain` SPARQL Neptune.

Cara menggunakan **`explain`** SPARQL untuk menganalisis eksekusi kueri Neptune

Fitur `explain` SPARQL adalah alat swalayan di Amazon Neptune yang membantu Anda memahami pendekatan eksekusi yang diambil oleh mesin Neptune. Untuk meminta `explain`, Anda melewatkan parameter ke permintaan HTTP atau HTTPS dalam bentuk `explain=`*`mode`*.

Nilai mode dapat menjadi salah satu dari `static`, `dynamic`, atau `details`:

- Dalam mode statis, `explain` mencetak hanya struktur statis dari rencana kueri.
- Dalam mode dinamis, `explain` juga mencakup aspek dinamis dari rencana kueri. Aspek-aspek ini mungkin menyertakan jumlah binding menengah mengalir melalui operator, rasio binding masuk ke binding keluar, dan total waktu yang dibutuhkan oleh operator.
- Dalam mode detail, `explain` mencetak informasi yang ditampilkan di mode `dynamic` ditambah rincian tambahan seperti string kueri SPARQL aktual dan perkiraan jumlah rentang untuk pola yang mendasari operator gabungan.

Neptune mendukung penggunaan `explain` dengan semua tiga protokol akses kueri SPARQL yang tercantum dalam spesifikasi Protokol SPARQL 1.1 W3C, yaitu:

1. HTTP GET
2. HTTP POST menggunakan parameter dikodekan URL
3. HTTP POST menggunakan parameter teks

Untuk informasi tentang mesin kueri SPARQL, lihat Memahami bagaimana mesin kueri SPARQL Neptune bekerja..

Untuk informasi tentang jenis output yang dihasilkan dengan menerapkan `explain` SPARQL, lihat Contoh `explain` SPARQL yang dipanggil di Neptune.

Contoh **`explain`** SPARQL yang dipanggil di Neptune

Contoh dalam bagian ini menunjukkan berbagai jenis output Anda dapat hasilkan dengan memanggil fitur `explain` SPARQL untuk menganalisis eksekusi kueri di Amazon Neptune.

Topik

- Memahami Output Explain
- Contoh output mode rincian
- Contoh dari output mode statis
- Berbagai cara pengkodean parameter
- Jenis output lain selain text/plain
- Contoh output explain SPARQL saat DFE diaktifkan

Memahami Output Explain

Dalam contoh ini, Jane Doe mengenal dua orang, yaitu John Doe dan Richard Roe:

```
@prefix ex: <http://example.com> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

ex:JaneDoe foaf:knows ex:JohnDoe .
ex:JohnDoe foaf:firstName "John" .
ex:JohnDoe foaf:lastName "Doe" .
```

```
ex:JaneDoe foaf:knows ex:RichardRoe .
ex:RichardRoe foaf:firstName "Richard" .
ex:RichardRoe foaf:lastName "Roe" .
.
```

Untuk menentukan nama depan semua orang yang diketahui Jane Doe, Anda dapat menulis kueri berikut:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -H "Accept: text/csv"
```

Kueri sederhana ini mengembalikan hal berikut ini:

```
firstName
John
Richard
```

Berikutnya, mengubah perintah `curl` untuk memangil `explain` dengan menambahkan `-d "explain=dynamic"` dan menggunakan tipe output default alih-alih `text/csv`:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=dynamic"
```

Kueri sekarang mengembalikan output dalam format ASCII pretty-printed (jenis konten HTTP `text/plain`), yang merupakan tipe output standar:

```
########################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     # Units In # Units Out # Ratio # Time (ms) #
########################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        # 0         # 1         # 0.00  # 0         #
```

```
#########################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        # 1        # 2         # 2.00  # 1         #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person]
            #          #          #           #       #           #
#########################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        # 2        # 2         # 1.00  # 1         #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person, ?firstName]
            #          #          #           #       #           #
#########################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   # 2        # 2         # 1.00  # 0         #
#########################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value # 2        # 2         # 1.00  # 1         #
#########################################################################################
```

Untuk rincian tentang operasi di kolom `Name` dan argumennya, lihat Operator explain.

Hal berikut ini menjelaskan output baris demi baris:

1. Langkah pertama dalam kueri utama selalu menggunakan operator `SolutionInjection` untuk menyuntikkan solusi. Solusinya kemudian diperluas ke hasil akhir melalui proses evaluasi.

   Dalam hal ini, operator menyuntikkan apa yang disebut solusi universal `{ }`. Dengan adanya klausa `VALUES` atau `BIND`, langkah ini mungkin juga menyuntikkan binding variabel yang lebih kompleks untuk memulai.

   Kolom `Units Out` menunjukkan bahwa solusi tunggal ini mengalir keluar dari operator. Kolom `Out #1` menentukan operator tempat operator ini mengumpankan hasilnya. Dalam contoh ini, semua operator terhubung ke operator yang mengikuti dalam tabel.
2. Langkah kedua adalah `PipelineJoin`. Operator menerima sebagai input solusi universal tunggal (sepenuhnya tidak dibatasi) yang dihasilkan oleh operator sebelumnya (`Units In := 1`). Operator menggabungkannya terhadap pola triple yang didefinisikan oleh argumen `pattern`. Hal ini terkait dengan pencarian sederhana untuk pola. Dalam kasus ini, pola triple didefinisikan sebagai berikut:

```
distinct( ex:JaneDoe, foaf:knows, ?person )
```

Argumen `joinType := join` menunjukkan bahwa ini adalah gabungan normal (jenis lain termasuk gabungan `optional`, gabungan `existence check`, dan sebagainya).

Argumen `distinct := true` mengatakan bahwa Anda mengekstrak hanya kecocokan khas dari database (tidak ada duplikat), dan Anda mengikat kecocokan khas ke variabel `joinProjectionVars := ?person`, dideduplikasi.

Fakta bahwa nilai kolom `Units Out` adalah 2 menunjukkan bahwa ada dua solusi yang mengalir keluar. Secara khusus, ini adalah binding untuk variabel `?person`, mencerminkan dua orang yang ditunjukkan data yang dikenal Jane Doe:

```
?person
-------------
ex:JohnDoe
ex:RichardRoe
```

3. Dua solusi dari tahap 2 mengalir sebagai input (`Units In := 2`) ke `PipelineJoin` yang kedua. Operator ini bergabung dengan dua solusi sebelumnya dengan pola triple berikut:

   ```
   distinct(?person, foaf:firstName, ?firstName)
   ```

   Variabel `?person` dikenal terikat baik ke `ex:JohnDoe` atau ke `ex:RichardRoe` oleh solusi masuk operator. Dengan begitu, `PipelineJoin` mengekstrak nama pertama, John dan Richard. Dua solusi keluar (Unit Out: = 2) kemudian sebagai berikut:

   ```
   ?person       | ?firstName
   ---------------------------
   ex:JohnDoe    | John
   ex:RichardRoe | Richard
   ```

4. Operator proyeksi berikutnya mengambil sebagai masukan dua solusi dari tahap 3 (`Units In := 2`) dan memproyeksikan ke variabel `?firstName`. Ini menghilangkan semua binding variabel lainnya dalam pemetaan dan melewati pada dua binding (`Units Out := 2`):

   ```
   ?firstName
   ----------
   John
   ```

```
  Richard
```

5. Untuk meningkatkan kinerja, Neptunus beroperasi jika memungkinkan pada pengidentifikasi internal yang ditetapkan untuk istilah URIs seperti dan string literal, bukan pada string itu sendiri. Operator terakhir, `TermResolution`, melakukan pemetaan dari pengidentifikasi internal ini kembali ke string istilah yang sesuai.

   Dalam evaluasi kueri biasa (non-explain), hasil dihitung oleh operator terakhir kemudian diserialisasi ke dalam format serialisasi yang diminta dan dialirkan ke klien.

Contoh output mode rincian

Misalkan Anda menjalankan kueri yang sama seperti sebelumnya di mode detail alih-alih mode dinamis:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=details"
```

Seperti yang ditunjukkan contoh ini, outputnya sama dengan beberapa rincian tambahan seperti string kueri di bagian atas output, dan `patternEstimate` bernilai untuk operator `PipelineJoin`:

```
Query:
PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://www.example.com/>
SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person foaf:firstName ?
firstName }

################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     # Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        # 0         # 1         # 0.00  # 0         #
################################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        # 1         # 2         # 2.00  # 13        #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
```

```
#    #        #        #                  # joinProjectionVars=[?person]
            #          #          #           #       #           #
#    #        #        #                  # patternEstimate=2
            #          #          #           #       #           #
#########################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        # 2        # 2         # 1.00  # 3         #
#    #        #        #                  # joinType=join
            #          #          #           #       #           #
#    #        #        #                  # joinProjectionVars=[?person, ?firstName]
            #          #          #           #       #           #
#    #        #        #                  # patternEstimate=2
            #          #          #           #       #           #
#########################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   # 2        # 2         # 1.00  # 1         #
#########################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value # 2        # 2         # 1.00  # 7         #
#########################################################################################
```

Contoh dari output mode statis

Misalkan Anda menjalankan kueri yang sama seperti sebelumnya di mode statis (default) alih-alih mode detail:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=static"
```

Seperti yang contoh ini tunjukkan, outputnya sama, kecuali bahwa itu menghilangkan tiga kolom terakhir:

```
#########################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     #
#########################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        #
```

```
##############################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        #
#    #        #        #                   # joinType=join
            #          #
#    #        #        #                   # joinProjectionVars=[?person]
            #          #
##############################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        #
#    #        #        #                   # joinType=join
            #          #
#    #        #        #                   # joinProjectionVars=[?person, ?firstName]
            #          #
##############################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   #
##############################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value #
##############################################################################################
```

Berbagai cara pengkodean parameter

Contoh kueri berikut menggambarkan dua cara yang berbeda untuk mengkodekan parameter ketika memanggil `explain` SPARQL.

Menggunakan pengkodean URL - Contoh ini menggunakan pengkodean URL parameter, dan menentukan output dinamis:

```
curl -XGET "http(s)://your_server:your_port/sparql?query=SELECT%20*%20WHERE%20%7B
%20%3Fs%20%3Fp%20%3Fo%20%7D%20LIMIT%20%31&explain=dynamic"
```

Menentukan parameter secara langsung — Ini sama dengan kueri sebelumnya kecuali bahwa ia melewatkan parameter melalui POST langsung:

```
curl http(s)://your_server:your_port/sparql \
  -d "query=SELECT * WHERE { ?s ?p ?o } LIMIT 1" \
  -d "explain=dynamic"
```

Jenis output lain selain text/plain

Contoh sebelumnya menggunakan jenis output `text/plain` default. Neptune juga dapat memformat output `explain` SPARQL dalam dua format jenis MIME lainnya, yaitu `text/csv` dan `text/html`. Anda memanggil mereka dengan menetapkan header `Accept` HTTP, yang dapat Anda lakukan menggunakan bendera `-H` di `curl`, sebagai berikut:

```
  -H "Accept: output type"
```

Berikut ini adalah beberapa contohnya:

Output **`text/csv`**

Kueri ini memanggil output jenis MIME CSV dengan menentukan `-H "Accept: text/csv"`:

```
curl http(s)://your_server:your_port/sparql \
  -d "query=SELECT * WHERE { ?s ?p ?o } LIMIT 1" \
  -d "explain=dynamic" \
  -H "Accept: text/csv"
```

Format CSV, yang berguna untuk diimport ke dalam spreadsheet atau basis data, memisahkan bidang dalam setiap baris `explain` dengan titik koma (`;`), seperti ini:

```
ID;Out #1;Out #2;Name;Arguments;Mode;Units In;Units Out;Ratio;Time (ms)
0;1;-;SolutionInjection;solutions=[{}];-;0;1;0.00;0
1;2;-;PipelineJoin;pattern=distinct(?s, ?p, ?o),joinType=join,joinProjectionVars=[?s, ?
p, ?o];-;1;6;6.00;1
2;3;-;Projection;vars=[?s, ?p, ?o];retain;6;6;1.00;2
3;-;-;Slice;limit=1;-;1;1;1.00;1
```

Output **`text/html`**

Jika Anda menentukan `-H "Accept: text/html"`, maka `explain` menghasilkan tabel HTML:

```
<!DOCTYPE html>
<html>
  <body>
    <table border="1px">
      <thead>
        <tr>
```

```
          <th>ID</th>
          <th>Out #1</th>
          <th>Out #2</th>
          <th>Name</th>
          <th>Arguments</th>
          <th>Mode</th>
          <th>Units In</th>
          <th>Units Out</th>
          <th>Ratio</th>
          <th>Time (ms)</th>
        </tr>
      </thead>

      <tbody>
        <tr>
          <td>0</td>
          <td>1</td>
          <td>-</td>
          <td>SolutionInjection</td>
          <td>solutions=[{}]</td>
          <td>-</td>
          <td>0</td>
          <td>1</td>
          <td>0.00</td>
          <td>0</td>
        </tr>

        <tr>
          <td>1</td>
          <td>2</td>
          <td>-</td>
          <td>PipelineJoin</td>
          <td>pattern=distinct(?s, ?p, ?o)<br>
              joinType=join<br>
              joinProjectionVars=[?s, ?p, ?o]</td>
          <td>-</td>
          <td>1</td>
          <td>6</td>
          <td>6.00</td>
          <td>1</td>
        </tr>

        <tr>
          <td>2</td>
```

```
          <td>3</td>
          <td>-</td>
          <td>Projection</td>
          <td>vars=[?s, ?p, ?o]</td>
          <td>retain</td>
          <td>6</td>
          <td>6</td>
          <td>1.00</td>
          <td>2</td>
        </tr>

        <tr>
          <td>3</td>
          <td>-</td>
          <td>-</td>
          <td>Slice</td>
          <td>limit=1</td>
          <td>-</td>
          <td>1</td>
          <td>1</td>
          <td>1.00</td>
          <td>1</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

HTML merender dalam browser seperti berikut ini:

ID	Out #1	Out #2	Name	Arguments	Mode	Units In	Units Out	Ratio	Time (ms)
0	1	-	SolutionInjection	solutions=[{}]	-	0	1	0.00	0
1	2	-	PipelineJoin	pattern=distinct(?s, ?p, ?o) joinType=join joinProjectionVars=[?s, ?p, ?o]	-	1	6	6.00	1
2	3	-	Projection	vars=[?s, ?p, ?o]	retain	6	6	1.00	2
3	-	-	Slice	limit=1	-	1	1	1.00	1

Contoh output **`explain`** SPARQL saat DFE diaktifkan

Berikut ini adalah contoh output `explain` SPARQL ketika mesin kueri alternatif DFE Neptune diaktifkan:

```
########################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
```

```
 # Mode     # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]


 # -        # 0        # 1         # 0.00  # 0         #
#########################################################################################
# 1  # 2      # -      # HashIndexBuild    # solutionSet=solutionSet1


 # -        # 1        # 1         # 1.00  # 22        #
#    #        #        #                   # joinVars=[]


 #          #          #           #       #           #
#    #        #        #                   # sourceType=pipeline


 #          #          #           #       #           #
#########################################################################################
# 2  # 3      # -      # DFENode            # DFE Stats=


      # -        # 101      # 100       # 0.99  # 32        #
#    #        #        #                   # ====> DFE execution time (measured by
 DFEQueryEngine)

     #          #          #           #       #           #
#    #        #        #                   # accepted [micros]=127


 #          #          #           #       #           #
#    #        #        #                   # ready [micros]=2


 #          #          #           #       #           #
#    #        #        #                   # running [micros]=5627


 #          #          #           #       #           #
#    #        #        #                   # finished [micros]=0
```

```
 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # ===> DFE execution time (measured in
 DFENode)

      #          #          #           #       #           #
#    #        #        #                   # -> setupTime [ms]=1


 #          #          #           #       #           #
#    #        #        #                   # -> executionTime [ms]=14


 #          #          #           #       #           #
#    #        #        #                   # -> resultReadTime [ms]=0


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # ===> Static analysis statistics


 #          #          #           #       #           #
#    #        #        #                   # --> 35907 micros spent in parser.


 #          #          #           #       #           #
#    #        #        #                   # --> 7643 micros spent in range count
 estimation
```

```
      #           #           #            #       #           #
#     #         #         #                  # --> 2895 micros spent in value resolution


 #           #           #            #       #           #
#     #         #         #                  #


 #           #           #            #       #           #
#     #         #         #                  # --> 39974925 micros spent in optimizer
 loop

    #           #           #            #       #           #
#     #         #         #                  #


 #           #           #            #       #           #
#     #         #         #                  #


 #           #           #            #       #           #
#     #         #         #                  # DFEJoinGroupNode[ children={


  #           #           #            #       #           #
#     #         #         #                  #   DFEPatternNode[(?1, TERM[117442062], ?
2, ?3) . project DISTINCT[?1, ?2] {rangeCountEstimate=100},

  #           #           #            #       #           #
#     #         #         #                  #     OperatorInfoWithAlternative[


  #           #           #            #       #           #
#     #         #         #                  #       rec=OperatorInfo[


  #           #           #            #       #           #
#     #         #         #                  #         type=INCREMENTAL_PIPELINE_JOIN,


 #           #           #            #       #           #
#     #         #         #                  #
 costEstimates=OperatorCostEstimates[
```

```
                                    #          #          #           #       #
  #
#    #        #        #                    #
 costEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0002,comp=0.0000,mem=0],

                                   #          #          #           #       #
 #
#    #        #        #                    #
 worstCaseCostEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0002,comp=0.0000,mem=0]

                       #          #          #           #       #           #
#    #        #        #                    #       alt=OperatorInfo[


  #          #          #           #       #           #
#    #        #        #                    #         type=INCREMENTAL_HASH_JOIN,


 #          #          #           #       #           #
#    #        #        #                    #
 costEstimates=OperatorCostEstimates[

                                    #          #          #           #       #
  #
#    #        #        #                    #
 costEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0003,comp=0.0000,mem=3212],

                                 #          #          #           #       #           #
#    #        #        #                    #
 worstCaseCostEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0003,comp=0.0000,mem=32

                  #          #          #           #       #           #
#    #        #        #                    #   DFEPatternNode[(?1, TERM[150997262], ?
4, ?5) . project DISTINCT[?1, ?4] {rangeCountEstimate=100},

  #          #          #           #       #           #
#    #        #        #                    #     OperatorInfoWithAlternative[


  #          #          #           #       #           #
#    #        #        #                    #       rec=OperatorInfo[
```

```
  #          #           #            #       #            #
#    #        #        #                   #         type=INCREMENTAL_HASH_JOIN,


 #          #           #            #       #            #
#    #        #        #                   #
 costEstimates=OperatorCostEstimates[

                                    #          #          #           #       #
  #
#    #        #        #                   #
 costEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0003,comp=0.0000,mem=6400],

                               #          #          #           #       #           #
#    #        #        #                   #
 worstCaseCostEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0003,comp=0.0000,mem=

                  #          #          #           #       #           #
#    #        #        #                   #       alt=OperatorInfo[


  #          #           #            #       #            #
#    #        #        #                   #         type=INCREMENTAL_PIPELINE_JOIN,


 #          #           #            #       #            #
#    #        #        #                   #
 costEstimates=OperatorCostEstimates[

                                    #          #          #           #       #
  #
#    #        #        #                   #
 costEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0010,comp=0.0000,mem=0],

                                #          #          #           #       #           #
#    #        #        #                   #
 worstCaseCostEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0010,comp=0.0000,mem=

                    #          #          #           #       #           #
#    #        #        #                   # },


 #          #           #            #       #            #
```

```
#    #        #        #                # ]


 #          #          #          #       #           #
#    #        #        #                #


 #          #          #          #       #           #
#    #        #        #                # ===> DFE configuration:


 #          #          #          #       #           #
#    #        #        #                # solutionChunkSize=5000


 #          #          #          #       #           #
#    #        #        #                # ouputQueueSize=20


 #          #          #          #       #           #
#    #        #        #                # numComputeCores=3


 #          #          #          #       #           #
#    #        #        #                # maxParallelIO=10


 #          #          #          #       #           #
#    #        #        #                # numInitialPermits=12


 #          #          #          #       #           #
#    #        #        #                #


 #          #          #          #       #           #
#    #        #        #                #


 #          #          #          #       #           #
#    #        #        #                # ====> DFE configuration (reported back)


 #          #          #          #       #           #
```

```
#    #        #        #                  # numComputeCores=3


 #          #          #           #       #           #
#    #        #        #                  # maxParallelIO=2


 #          #          #           #       #           #
#    #        #        #                  # numInitialPermits=12


 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # ===> Statistics & operator histogram


 #          #          #           #       #           #
#    #        #        #                  # ==> Statistics


 #          #          #           #       #           #
#    #        #        #                  # -> 3741 / 3668 micros total elapsed (incl.
 wait / excl. wait)

 #          #          #           #       #           #
#    #        #        #                  # -> 3741 / 3 millis total elapse (incl.
 wait / excl. wait)

    #          #          #           #       #           #
#    #        #        #                  # -> 3741 / 0 secs total elapsed (incl.
 wait / excl. wait)

     #          #          #           #       #           #
#    #        #        #                  # ==> Operator histogram


 #          #          #           #       #           #
#    #        #        #                  # -> 47.66% of total time (excl. wait):
 pipelineScan (2 instances)

     #          #          #           #       #           #
```

```
#    #        #        #                  # -> 10.99% of total time (excl. wait):
 merge (1 instances)

     #          #          #           #       #           #
#    #        #        #                  # -> 41.17% of total time (excl. wait):
 symmetricHashJoin (1 instances)

     #          #          #           #       #           #
#    #        #        #                  # -> 0.19% of total time (excl. wait): drain
 (1 instances)

 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # nodeId | out0   | out1 | opName
   | args                                          | rowsIn | rowsOut | chunksIn |
 chunksOut | elapsed* | outWait | outBlocked | ratio    | rate* [M/s] | rate [M/s] | %
    #          #          #           #       #           #
#    #        #        #                  # ------ | ------ | ---- | -----------------
 | ---------------------------------------------- | ------ | ------- | -------- |
 --------- | -------- | ------- | ---------- | -------- | ----------- | ---------- |
 ----- #          #          #           #       #           #
#    #        #        #                  # node_0 | node_2 | -    | pipelineScan
  | (?1, TERM[117442062], ?2, ?3) DISTINCT [?1, ?2]  | 0      | 100     | 0        | 1
        | 874      | 0       | 0          | Infinity | 0.1144      | 0.1144     | 23.83
 #          #          #           #       #           #
#    #        #        #                  # node_1 | node_2 | -    | pipelineScan
  | (?1, TERM[150997262], ?4, ?5) DISTINCT [?1, ?4]  | 0      | 100     | 0        | 1
        | 874      | 0       | 0          | Infinity | 0.1144      | 0.1144     | 23.83
 #          #          #           #       #           #
#    #        #        #                  # node_2 | node_4 | -    | symmetricHashJoin
 |                                                  | 200    | 100     | 2        | 2
        | 1510     | 73      | 0          | 0.50     | 0.0662      | 0.0632     | 41.17
 #          #          #           #       #           #
#    #        #        #                  # node_3 | -      | -    | drain
  |                                                  | 100    | 0       | 1        | 0
        | 7        | 0       | 0          | 0.00     | 0.0000      | 0.0000     | 0.19
 #          #          #           #       #           #
#    #        #        #                  # node_4 | node_3 | -    | merge
  |                                                  | 100    | 100     | 2        | 1
        | 403      | 0       | 0          | 1.00     | 0.2481      | 0.2481     | 10.99
 #          #          #           #       #           #
```

```
##############################################################################################
# 3  # 4      # -      # HashIndexJoin     # solutionSet=solutionSet1


 # -        # 100      # 100       # 1.00  # 4         #
#    #        #        #                   # joinType=join


 #          #          #           #       #           #
##############################################################################################
# 4  # 5      # -      # Distinct          # vars=[?s, ?o, ?o1]


 # -        # 100      # 100       # 1.00  # 9         #
##############################################################################################
# 5  # 6      # -      # Projection        # vars=[?s, ?o, ?o1]


 # retain   # 100      # 100       # 1.00  # 2         #
##############################################################################################
# 6  # -      # -      # TermResolution    # vars=[?s, ?o, ?o1]


 # id2value # 100      # 100       # 1.00  # 11        #
##############################################################################################
```

Operator **explain** SPARQL Neptune

Bagian berikut menjelaskan operator dan parameter untuk fitur `explain` SPARQL yang saat ini tersedia di Amazon Neptune.

> **Important**
>
> Fitur `explain` SPARQL sedang disempurnakan. Operator dan parameter yang didokumentasikan di sini mungkin berubah dalam versi mendatang.

Topik

- Operator Aggregation
- Operator ConditionalRouting
- Operator Copy

- Operator DFENode
- Operator Distinct
- Operator Federation
- Operator Filter
- Operator HashIndexBuild
- Operator HashIndexJoin
- Operator MergeJoin
- Operator NamedSubquery
- Operator PipelineJoin
- Operator PipelineCountJoin
- Operator PipelinedHashIndexJoin
- Operator Projection
- Operator PropertyPath
- Operator TermResolution
- Operator Slice
- Operator SolutionInjection
- Operator Sort
- Operator VariableAlignment

Operator **`Aggregation`**

Melakukan satu agregasi atau lebih, menerapkan semantik dari operator agregasi SPARQL seperti `count`, `max`, `min`, `sum`, dan sebagainya.

`Aggregation` dilengkapi dengan pengelompokan opsional menggunakan klausa `groupBy`, dan kendala `having` opsional.

Pendapat

- `groupBy` — (Opsional) Menyediakan klaus `groupBy` yang menentukan urutan ekspresi yang menurut pengelompokan solusi masuk.
- `aggregates` — (Wajib) Menentukan daftar urut ekspresi agregasi.

- `having` — (Opsional) Menambahkan kendala untuk menyaring grup, seperti yang disiratkan oleh klausul `having` dalam kueri SPARQL.

Operator **`ConditionalRouting`**

Merutekan solusi masuk berdasarkan kondisi tertentu. Solusi yang memenuhi kondisi diarahkan ke ID operator yang direferensikan oleh `Out #1`, sedangkan solusi yang tidak diarahkan ke operator direferensikan oleh `Out #2`.

Pendapat

- `condition` — (Wajib) Kondisi perutean.

Operator **`Copy`**

Mendelegasikan aliran solusi sebagaimana ditentukan oleh mode tertentu.

Mode

- `forward` — Meneruskan solusi untuk operator hilir yang diidentifikasi oleh `Out #1`.
- `duplicate` — Menduplikat solusi dan meneruskannya ke masing-masing dari dua operator yang diidentifikasi oleh `Out #1` dan `Out #2`.

`Copy` tidak memiliki argumen.

Operator **`DFENode`**

Operator ini merupakan abstraksi dari rencana yang dijalankan oleh mesin kueri alternatif DFE. Rencana DFE rinci diuraikan dalam argumen untuk operator ini. Argumen saat ini kelebihan beban untuk berisi statistik runtime rinci dari rencana DFE. Argumen berisi waktu yang dihabiskan dalam berbagai langkah eksekusi kueri oleh DFE.

Abstrak sintaks tree (AST) yang dioptimalkan logis untuk rencana kueri DFE dicetak dengan informasi tentang jenis operator yang dianggap saat perencanaan dan akibat terbaik dan terburuk terkait untuk menjalankan operator. AST saat ini terdiri dari jenis node berikut:

- `DFEJoinGroupNode` — Merupakan gabungan dari satu `DFEPatternNodes` atau lebih.
- `DFEPatternNode` - Merangkum pola mendasar menggunakan tupel yang cocok yang diproyeksikan keluar dari database yang mendasari.

Sub-bagiannya, `Statistics & Operator histogram`, berisi rincian tentang waktu eksekusi rencana `Dataflow0p` dan rincian waktu CPU yang digunakan oleh masing-masing operator. Di bawah ini ada tabel yang mencetak statistik runtime rinci dari rencana yang dieksekusi oleh DFE.

> **Note**
>
> Karena DFE adalah fitur eksperimental yang dirilis dalam mode lab, format tepat output `explain` dapat berubah.

Operator **`Distinct`**

Menghitung proyeksi yang berbeda pada subset dari variabel, menghilangkan duplikat. Akibatnya, jumlah solusi yang mengalir lebih besar dari atau sama dengan jumlah solusi yang mengalir keluar.

Pendapat

- `vars` — (Wajib) Variabel tempat proyeksi `Distinct` akan diterapkan.

Operator **`Federation`**

Melewatkan kueri tertentu ke titik akhir SPARQL remote yang ditentukan.

Pendapat

- `endpoint` — (Wajib) URL titik akhir dalam pernyataan `SERVICE` SPARQL. Ini bisa berupa string konstan, atau jika titik akhir kueri ditentukan berdasarkan variabel dalam kueri yang sama, ia dapat menjadi nama variabel.
- `query` — (Wajib) String kueri yang direkonstruksi yang akan dikirim ke titik akhir remote. Mesin menambahkan prefiks default untuk kueri ini bahkan ketika klien tidak menentukan apapun.
- `silent` — (Wajib) Boolean yang menunjukkan apakah kata kunci `SILENT` muncul setelah kata kunci tersebut. `SILENT` memberi tahu mesin untuk tidak menggagalkan seluruh kueri bahkan jika bagian `SERVICE` remote gagal.

Operator **`Filter`**

Menyaring solusi yang masuk. Hanya solusi yang memenuhi kondisi filter yang diteruskan ke operator hulu, dan yang lainnya dijatuhkan.

Pendapat

- `condition` — (Wajib) Kondisi filter.

Operator **`HashIndexBuild`**

Membawa daftar binding dan melakukan spools pada mereka ke dalam indeks hash yang namanya didefinisikan oleh argumen `solutionSet`. Biasanya, operator berikutnya melakukan penggabungan terhadap set solusi ini, mengacunya dengan nama itu.

Pendapat

- `solutionSet` — (Wajib) Nama set solusi indeks hash.
- `sourceType` — (Wajib) Jenis sumber asal binding yang akan disimpan dalam indeks hash diperoleh:
  - `pipeline` — Melakukan spool pada solusi yang masuk dari operator hilir dalam alur operator ke dalam indeks hash.
  - `binding set` - Melakukan spool pada set mengikat tetap yang ditentukan oleh argumen `sourceBindingSet` ke dalam indeks hash.
- `sourceBindingSet` — (Opsional) Jika nilai argumen `sourceType` adalah `binding set`, argumen ini menentukan set binding statis yang akan di-spool ke dalam indeks hash.

Operator **`HashIndexJoin`**

Menggabungkan solusi masuk terhadap set solusi indeks hash yang diidentifikasi oleh argumen `solutionSet`.

Pendapat

- `solutionSet` — (Wajib) Nama set solusi yang akan digabungkan. Ini harus berupa indeks hash yang telah dibangun di langkah sebelumnya menggunakan operator `HashIndexBuild`.
- `joinType` — (Wajib) Jenis gabungan yang akan dilakukan:
  - `join` — Gabungan normal, membutuhkan kecocokan persis antara semua variabel bersama.
  - `optional` — Sebuah gabungan `optional` yang menggunakan semantik operator `OPTIONAL` SPARQL.
  - `minus` — Sebuah operasi `minus` yang mempertahankan pemetaan yang tanpa mitra gabungan, menggunakan semanti operator `MINUS` SPARQL.

  - `existence check` — Memeriksa apakah ada mitra gabungan atau tidak, dan mengikat variabel `existenceCheckResultVar` ke hasil pemeriksaan ini.
- `constraints` — (Opsional) Tambahan kendala gabungan yang dipertimbangkan selama penggabungan. Gabungan yang tidak memenuhi kendala ini dibuang.
- `existenceCheckResultVar` — (Opsional) Hanya digunakan untuk gabungan di mana `joinType` sama dengan `existence check` (lihat argumen `joinType` sebelumnya).

Operator **`MergeJoin`**

Penggabungan yang digabung di lebih dari beberapa set solusi, seperti yang diidentifikasi oleh argumen `solutionSets`.

Pendapat

- `solutionSets` — (Wajib) Set solusi yang akan digabungkan bersama.

Operator **`NamedSubquery`**

Memicu evaluasi subkueri yang diidentifikasi oleh argumen `subQuery` dan melakukan spools pada hasil ke dalam set solusi yang ditentukan oleh argumen `solutionSet`. Solusi masuk untuk operator diteruskan ke subkueri dan kemudian ke operator berikutnya.

Pendapat

- `subQuery` — (Wajib) Nama subkueri yang akan dievaluasi. Subkueri dirender secara eksplisit dalam output.
- `solutionSet` — (Wajib) Nama set solusi tempat menyimpan hasil subkueri.

Operator **`PipelineJoin`**

Menerima output dari operator sebelumnya sebagai input dan menggabungkannya dengan pola tuple yang didefinisikan oleh argumen `pattern`.

Pendapat

- `pattern`— (Wajib) Pola, yang mengambil bentuk subject-predicate-object, dan tupel grafik opsional yang mendasari gabungan. Jika `distinct` ditentukan untuk pola, gabungan hanya

mengekstrak solusi yang khas dari variabel proyeksi yang ditentukan oleh `projectionVars`, alih-alih semua solusi yang cocok.

- `inlineFilters` — (Opsional) Satu set filter yang akan diterapkan pada variabel dalam pola. Pola dievaluasi bersamaan dengan filter ini.
- `joinType` — (Wajib) Jenis gabungan yang akan dilakukan:
  - `join` — Gabungan normal, membutuhkan kecocokan persis antara semua variabel bersama.
  - `optional` — Sebuah gabungan `optional` yang menggunakan semantik operator `OPTIONAL` SPARQL.
  - `minus` — Sebuah operasi `minus` yang mempertahankan pemetaan yang tanpa mitra gabungan, menggunakan semanti operator `MINUS` SPARQL.
  - `existence check` — Memeriksa apakah ada mitra gabungan atau tidak, dan mengikat variabel `existenceCheckResultVar` ke hasil pemeriksaan ini.
- `constraints` — (Opsional) Tambahan kendala gabungan yang dipertimbangkan selama penggabungan. Gabungan yang tidak memenuhi kendala ini dibuang.
- `projectionVars`— (Opsional) Variabel proyeksi. Digunakan dalam kombinasi bersama `distinct := true` untuk menegakkan ekstraksi proyeksi khas pada satu set variabel tertentu.
- `cutoffLimit` — (Opsional) Batas cutoff untuk jumlah mitra gabungan yang diekstrak. Meskipun tidak ada batas secara default, Anda dapat mengatur ini ke 1 ketika melakukan gabungan untuk menerapkan klausa `FILTER (NOT) EXISTS`, di mana itu cukup untuk membuktikan atau membantah bahwa ada mitra gabungan.

Operator **`PipelineCountJoin`**

Varian dari `PipelineJoin`. Alih-alih bergabung, operator ini hanya menghitung mitra gabungan yang cocok dan mengikat hitungan ke variabel yang ditentukan oleh argumen `countVar`.

Pendapat

- `countVar` — (Wajib) Variabel yang akan diikatkan pada hasil hitungan, yaitu jumlah mitra gabungan,.
- `pattern`— (Wajib) Pola, yang mengambil bentuk subject-predicate-object, dan tupel grafik opsional yang mendasari gabungan. Jika `distinct` ditentukan untuk pola, gabungan hanya mengekstrak solusi yang khas dari variabel proyeksi yang ditentukan oleh `projectionVars`, alih-alih semua solusi yang cocok.

- `inlineFilters` — (Opsional) Satu set filter yang akan diterapkan pada variabel dalam pola. Pola dievaluasi bersamaan dengan filter ini.
- `joinType` — (Wajib) Jenis gabungan yang akan dilakukan:
  - `join` — Gabungan normal, membutuhkan kecocokan persis antara semua variabel bersama.
  - `optional` — Sebuah gabungan `optional` yang menggunakan semantik operator `OPTIONAL` SPARQL.
  - `minus` — Sebuah operasi `minus` yang mempertahankan pemetaan yang tanpa mitra gabungan, menggunakan semanti operator `MINUS` SPARQL.
  - `existence check` — Memeriksa apakah ada mitra gabungan atau tidak, dan mengikat variabel `existenceCheckResultVar` ke hasil pemeriksaan ini.
- `constraints` — (Opsional) Tambahan kendala gabungan yang dipertimbangkan selama penggabungan. Gabungan yang tidak memenuhi kendala ini dibuang.
- `projectionVars`— (Opsional) Variabel proyeksi. Digunakan dalam kombinasi bersama `distinct := true` untuk menegakkan ekstraksi proyeksi khas pada satu set variabel tertentu.
- `cutoffLimit` — (Opsional) Batas cutoff untuk jumlah mitra gabungan yang diekstrak. Meskipun tidak ada batas secara default, Anda dapat mengatur ini ke 1 ketika melakukan gabungan untuk menerapkan klausa `FILTER (NOT) EXISTS`, di mana itu cukup untuk membuktikan atau membantah bahwa ada mitra gabungan.

Operator **`PipelinedHashIndexJoin`**

Ini adalah indeks hash all-in-one build dan operator gabungan. Dibutuhkan daftar binding, menggulungnya menjadi indeks hash, dan kemudian bergabung dengan solusi yang masuk terhadap indeks hash.

Pendapat

- `sourceType`— (Wajib) Jenis sumber dari mana binding untuk menyimpan dalam indeks hash diperoleh, salah satu dari:
  - `pipeline`— Penyebab `PipelinedHashIndexJoin` untuk menggeser solusi yang masuk dari operator hilir di pipa operator ke indeks hash.
  - `binding set`— Penyebab `PipelinedHashIndexJoin` untuk menggeser set pengikatan tetap yang ditentukan oleh `sourceBindingSet` argumen ke dalam indeks hash.
- `sourceSubQuery` — (Opsional) Jika nilai `sourceType` argumen adalah`pipeline`, argumen ini menentukan subquery yang dievaluasi dan spooled ke dalam indeks hash.

- `sourceBindingSet` — (Opsional) Jika nilai `sourceType` argumen adalah`binding set`, argumen ini menentukan set pengikatan statis yang akan di-spooled ke indeks hash.
- `joinType`— (Wajib) Jenis bergabung yang akan dilakukan:
  - `join` — Gabungan normal, membutuhkan kecocokan persis antara semua variabel bersama.
  - `optional` — Sebuah gabungan `optional` yang menggunakan semantik operator `OPTIONAL` SPARQL.
  - `minus` — Sebuah operasi `minus` yang mempertahankan pemetaan yang tanpa mitra gabungan, menggunakan semanti operator `MINUS` SPARQL.
  - `existence check` — Memeriksa apakah ada mitra gabungan atau tidak, dan mengikat variabel `existenceCheckResultVar` ke hasil pemeriksaan ini.
- `existenceCheckResultVar`— (Opsional) Hanya digunakan untuk bergabung di mana `joinType` sama `existence check` (lihat argumen JoinType di atas).

Operator **`Projection`**

Memproyeksikan di atas subset variabel. Jumlah solusi yang mengalir masuk sama dengan jumlah solusi yang mengalir keluar, namun bentuk solusinya berbeda, tergantung pengaturan mode.

Mode

- `retain` — Menyimpan dalam solusi hanya variabel yang ditentukan oleh argumen `vars`.
- `drop` — Jatuhkan semua variabel yang ditentukan oleh argumen `vars`.

Pendapat

- `vars`— (Wajib) Variabel yang akan dipertahankan atau dijatuhkan, tergantung pengaturan mode.

Operator **`PropertyPath`**

Memungkinkan jalur properti rekursif seperti `+` atau `*`. Neptune mengimplementasikan pendekatan iterasi titik tetap berdasarkan template yang ditentukan oleh argumen `iterationTemplate`. Variabel sisi kiri atau kanan yang dikenal terikat dalam template untuk setiap iterasi titik tetap, sampai tidak ada lagi solusi baru yang dapat ditemukan.

Pendapat

- `iterationTemplate`— (Wajib) Nama dari template subkueri yang digunakan untuk mengimplementasikan iterasi titik tetap.
- `leftTerm`— (Wajib) Istilah (variabel atau konstan) di sisi kiri jalur properti.
- `rightTerm`— (Wajib) Istilah (variabel atau konstan) di sisi kanan jalur properti.
- `lowerBound`— (Wajib) Batas bawah untuk iterasi titik tetap (baik `0` untuk kueri `*`, atau `1` untuk kueri `+`).

Operator **`TermResolution`**

Menerjemahkan nilai-nilai pengidentifikasi string internal kembali ke string eksternal mereka yang sesuai, atau menerjemahkan string eksternal ke nilai-nilai pengidentifikasi string internal, tergantung modenya.

Mode

- `value2id`— Memetakan istilah seperti literal dan nilai ID internal yang URIs sesuai (pengkodean ke nilai internal).
- `id2value`— Memetakan nilai ID internal ke istilah yang sesuai seperti literal dan URIs (decoding nilai internal).

Pendapat

- `vars`— (Wajib) Menentukan variabel yang string atau string internal IDs harus dipetakan.

Operator **`Slice`**

Mengimplementasikan potongan di atas aliran solusi masuk, menggunakan semantik klausa `LIMIT` dan `OFFSET` SPARQL.

Pendapat

- `limit`— (Opsional) Batas pada solusi yang akan diteruskan.
- `offset`— (Opsional) Offset di mana solusi dievaluasi untuk penerusan.

Operator **`SolutionInjection`**

Tidak menerima input. Menyuntikkan solusi secara statis ke dalam rencana kueri dan mencatatnya dalam argumen `solutions`.

Rencana kueri selalu dimulai dengan injeksi statis ini. Jika solusi statis yang akan disuntikkan dapat berasal dari kueri itu sendiri dengan menggabungkan berbagai sumber binding statis (misalnya, dari klausa `VALUES` atau `BIND`), maka operator `SolutionInjection` menyuntikkan solusi statis turunan ini. Dalam kasus yang paling sederhana, hal ini mencerminkan binding yang tersirat oleh klausa `VALUES` luar.

Jika tidak ada solusi statis dapat diturunkan dari kueri, `SolutionInjection` menyuntikkan solusi kosong, yang disebut solusi universal, yang diperluas dan digandakan sepanjang proses evaluasi kueri.

Pendapat

- `solutions`— (Wajib) Urutan solusi yang disuntikkan oleh operator.

Operator **`Sort`**

Mengurutkan set solusi menggunakan kondisi pengurutan tertentu.

Pendapat

- `sortOrder`— (Wajib) Daftar variabel yang diurutkan, masing-masing berisi pengidentifikasi `ASC` (naik) atau `DESC` (turun), digunakan secara berurutan untuk mengurutkan set solusi.

Operator **`VariableAlignment`**

Memeriksa solusi satu per satu, melakukan penyelarasan pada masing-masing pada dua variabel: `sourceVar` yang ditentukan dan `targetVar` yang ditentukan.

Jika `sourceVar` dan `targetVar` dalam solusi memiliki nilai yang sama, variabel dianggap selaras dan solusi diteruskan, dengan `sourceVar` berulang diproyeksikan keluar.

Jika variabel terikat ke nilai yang berbeda, solusinya disaring seluruhnya.

Pendapat

- `sourceVar`— (Wajib) Variabel sumber, untuk dibandingkan dengan variabel target. Jika penyelarasan berhasil dalam solusi, yang berarti bahwa dua variabel memiliki nilai yang sama, variabel sumber diproyeksikan keluar.
- `targetVar`— (Wajib) Variabel target, sebagai pembanding variabel sumber. Dipertahankan bahkan ketika penyelarasan berhasil.

Keterbatasan **`explain`** SPARQL di Neptune

Rilis fitur `explain` dari SPARQL Neptune memiliki keterbatasan berikut.

Neptune Saat ini Mendukung Explain Hanya dalam Kueri SPARQL SELECT

Untuk informasi tentang proses evaluasi untuk bentuk kueri lainnya, seperti kueri `ASK`, `CONSTRUCT`, `DESCRIBE`, dan `SPARQL UPDATE`, Anda dapat mengubah kueri-kueri ini menjadi kueri SELECT. Kemudian gunakan `explain` untuk memeriksa kueri SELECT sesuai sebagai gantinya.

Misalnya, untuk mendapatkan Informasi `explain` tentang kueri `ASK WHERE {...}`, jalankan kueri `SELECT WHERE {...} LIMIT 1` yang sesuai bersama `explain`.

Demikian pula, untuk kueri `CONSTRUCT {...} WHERE {...}`, jatuhkan bagian `CONSTRUCT {...}` dan jalankan kueri `SELECT` dengan `explain` pada klausa `WHERE {...}` kedua. Mengevaluasi klausa `WHERE` kedua umumnya mengungkapkan tantangan utama memproses kueri `CONSTRUCT`, karena solusi yang mengalir keluar dari `WHERE` kedua ke dalam template `CONSTRUCT` umumnya hanya membutuhkan substitusi langsung.

Operator Explain Dapat Berubah di Rilis yang Akan Datang

Operator `explain` SPARQL dan parameternya dapat berubah dalam rilis yang akan datang.

Output Explain Dapat Berubah di Rilis yang Akan Datang

Misalnya, header kolom bisa berubah, dan lebih banyak kolom mungkin ditambahkan ke tabel.

Kueri gabungan SPARQL di Neptune menggunakan ekstensi **`SERVICE`**

Amazon Neptune sepenuhnya mendukung ekstensi kueri gabungan SPARQL yang menggunakan kata kunci `SERVICE`. (Untuk informasi selengkapnya, lihat Kueri Gabungan SPARQL 1.1.)

Kata kunci `SERVICE` menginstruksikan mesin kueri SPARQL untuk mengeksekusi sebagian dari kueri terhadap titik akhir SPARQL jauh dan menyusun hasil kueri akhir. Hanya operasi `READ` yang dimungkinkan. Operasi `WRITE` dan `DELETE` tidak didukung. Neptune hanya dapat menjalankan kueri gabungan terhadap titik akhir SPARQL yang dapat diakses dalam cloud privat virtual (VPC). Namun, Anda juga dapat menggunakan reverse proxy di VPC untuk membuat sumber data eksternal dapat diakses dalam VPC.

> **Note**
>
> Saat `SERVICE` SPARQL digunakan untuk melakukan federasi pada kueri untuk dua klaster Neptune atau lebih di VPC yang sama, grup keamanan harus dikonfigurasi untuk memungkinkan semua klaster Neptune untuk berbicara satu sama lain.

> **Important**
>
> SPARQL 1.1 Federation membuat permintaan layanan atas nama Anda saat melewatkan kueri dan parameter ke titik akhir SPARQL eksternal. Anda bertanggung jawab untuk memverifikasi bahwa titik akhir SPARQL eksternal memenuhi persyaratan penanganan data dan keamanan aplikasi Anda.

Contoh dari kueri gabungan Neptune

Contoh sederhana berikut menunjukkan bagaimana kueri gabungan SPARQL bekerja.

Misalkan pelanggan mengirimkan kueri berikut ke Neptune-1 di `http://neptune-1:8182/sparql`.

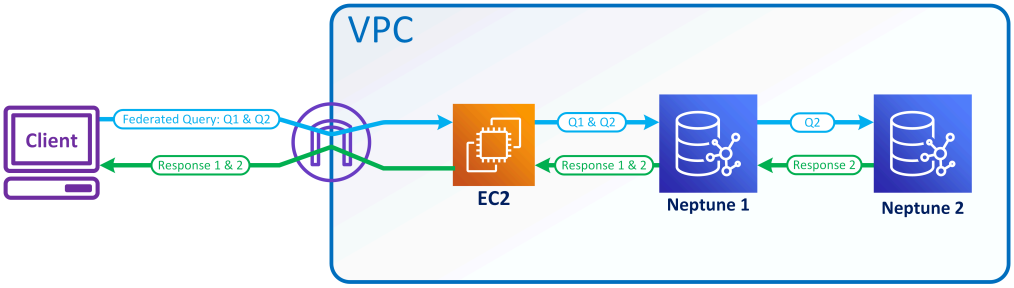
```
SELECT * WHERE {
   ?person rdf:type foaf:Person .
   SERVICE <http://neptune-2:8182/sparql> {
       ?person foaf:knows ?friend .
    }
}
```

1. Neptune-1 mengevaluasi pola kueri pertama (Q-1) yang merupakan `?person rdf:type foaf:Person`, menggunakan hasilnya untuk menyelesaikan `?person` di Q-2 (`?person`

`foaf:knows ?friend`), dan meneruskan pola yang dihasilkan ke Neptune-2 di `http://neptune-2:8182/sparql`.

2. Neptune-2 mengevaluasi Q-2 dan mengirimkan hasilnya kembali ke Neptune-1.
3. Neptune-1 menggabungkan solusi untuk kedua pola dan mengirimkan hasilnya kembali ke pelanggan.

Alur ini ditunjukkan dalam diagram berikut.



> **Note**
>
> “Secara default, optimizer menentukan pada titik apa dalam eksekusi kueri bahwa instruksi `SERVICE` dijalankan. Anda dapat mengganti penempatan ini menggunakan petunjuk kueri JoinOrder.

Kontrol akses untuk kueri gabungan di Neptune

Neptunus AWS Identity and Access Management menggunakan (IAM) untuk otentikasi dan otorisasi. Kontrol akses untuk kueri gabungan dapat melibatkan lebih dari satu instans DB Neptune. Instans ini mungkin memiliki persyaratan yang berbeda untuk kontrol akses. Dalam keadaan tertentu, ini dapat membatasi kemampuan Anda untuk membuat kueri gabungan.

Pertimbangkan contoh sederhana yang disajikan di bagian sebelumnya. Neptune-1 memanggil Neptune-2 dengan kredensial yang sama yang memanggilnya.

- Jika Neptune-1 memerlukan autentikasi dan otorisasi IAM, namun tidak untuk Neptune-2, yang Anda butuhkan hanya izin IAM yang sesuai untuk Neptune-1 untuk membuat kueri gabungan.
- Jika Neptune-1 dan Neptune-2 keduanya memerlukan autentikasi dan otorisasi IAM, Anda perlu melampirkan izin IAM untuk kedua database untuk membuat kueri gabungan. Kedua cluster juga harus berada di AWS akun yang sama dan di wilayah yang sama. Arsitektur kueri and/or federasi lintas wilayah lintas wilayah saat ini tidak didukung.

- Namun, dalam kasus di mana Neptune-1 IAM-nya tidak diaktifkan tetapi diaktifkan di Neptune-2, Anda tidak dapat membuat kueri gabungan. Alasannya adalah bahwa Neptune-1 tidak dapat mengambil kredensial IAM Anda dan meneruskannya ke Neptune-2 untuk mengotorisasi bagian kedua dari kueri.

Alat visualisasi grafik untuk Neptunus

Selain kemampuan visualisasi yang dibangun ke dalam grafik-notebook Neptunus, Anda juga dapat menggunakan solusi yang dibangun AWS oleh mitra dan vendor pihak ketiga untuk memvisualisasikan data yang disimpan di Neptunus.

Visualisasi grafik yang canggih dapat membantu ilmuwan data, manajer, dan peran lain dalam organisasi mengeksplorasi data grafik secara interaktif, tanpa harus tahu cara menulis kueri yang kompleks.

Topik

- Penjelajah grafik sumber terbuka
- Perangkat Lunak Tom Sawyer
- Kecerdasan Cambridge
- Grafis
- metafak
- Klien database grafik G.V ()
- Linkurious
- Graph.Build

Penjelajah grafik sumber terbuka

Graph-explorer adalah alat eksplorasi visual kode rendah sumber terbuka untuk data grafik, tersedia di bawah lisensi Apache-2.0. Ini memungkinkan Anda menelusuri data grafik properti berlabel (LPG) atau Resource Description Framework (RDF) dalam database grafik tanpa harus menulis kueri grafik. Graph-explorer dimaksudkan untuk membantu ilmuwan data, analis bisnis, dan peran lain dalam organisasi mengeksplorasi data grafik secara interaktif tanpa harus mempelajari bahasa kueri grafik.

Graph-explorer menyediakan aplikasi web berbasis React yang dapat digunakan sebagai wadah untuk memvisualisasikan data grafik. Anda dapat terhubung ke Amazon Neptunus atau ke database grafik lain yang menyediakan titik akhir TinkerPop Apache Gremlin atau SPARQL 1.1.

- Anda dapat dengan cepat melihat ringkasan data menggunakan filter faset, atau mencari data dengan mengetikkan teks ke dalam bilah pencarian.

- Anda juga dapat secara interaktif menjelajahi koneksi node dan edge. Anda dapat melihat tetangga simpul untuk melihat bagaimana objek berhubungan satu sama lain, dan kemudian menelusuri untuk memeriksa tepi dan properti secara visual.
- Anda juga dapat menyesuaikan tata letak grafik, warna, ikon, dan properti default mana yang akan ditampilkan untuk node dan tepi. Untuk grafik RDF, Anda juga dapat menyesuaikan ruang nama untuk sumber daya. URIs
- Untuk laporan dan presentasi yang melibatkan data grafik, Anda dapat mengonfigurasi dan menyimpan tampilan yang telah Anda buat dalam format PNG resolusi tinggi. Anda juga dapat mengunduh data terkait ke file CSV atau JSON untuk diproses lebih lanjut.

Menggunakan graph-explorer di notebook grafik Neptunus

Cara termudah untuk menggunakan graph-explorer dengan Neptunus adalah di notebook grafik Neptunus.

Jika Anda menggunakan meja kerja Neptunus untuk meng-host notebook Neptunus, graph-explorer secara otomatis digunakan dengan notebook dan terhubung ke Neptunus.

Setelah Anda membuat buku catatan, buka konsol Neptunus untuk memulai graph-explorer:

1. Pergi ke Neptunus.
2. Di bawah Notebook, pilih buku catatan Anda.
3. Di bawah Tindakan pilih Buka Penjelajah Grafik.

Cara menjalankan graph-explorer di Amazon ECS AWS Fargate dan terhubung ke Neptunus

Anda juga dapat membuat image Docker graph-explorer dan menjalankannya di mesin lokal atau layanan yang dihosting seperti Amazon Elastic Compute Cloud (Amazon EC2) atau Amazon Elastic Container Service (Amazon ECS), seperti yang dijelaskan di bagian Memulaidari read-me dalam proyek graph-explorer. GitHub

Sebagai contoh, bagian ini memberikan step-by-step instruksi untuk menjalankan graph-explorer di Amazon ECS pada: AWS Fargate

1. Buat peran IAM baru dan lampirkan kebijakan ini padanya:

- Amazon ECSTask ExecutionRolePolicy
- CloudWatchLogsFullAccess

Jaga agar nama peran berguna untuk digunakan dalam satu menit.

2. Buat klaster Amazon ECS dengan infrastruktur yang disetel ke FARGATE dan opsi jaringan berikut:

   - VPC: atur ke VPC tempat database Neptunus Anda berada.
   - Subnets: atur ke subnet publik VPC itu (hapus semua yang lain).

3. Buat definisi tugas JSON baru sebagai berikut:

```
{
  "family": "explorer-test",
  "containerDefinitions": [
    {
      "name": "graph-explorer",
      "image": "public.ecr.aws/neptune/graph-explorer:latest",
      "cpu": 0,
      "portMappings": [
        {
          "name": "graph-explorer-80-tcp",
          "containerPort": 80,
          "hostPort": 80,
          "protocol": "tcp",
          "appProtocol": "http"
        },
        {
          "name": "graph-explorer-443-tcp",
          "containerPort": 443,
          "hostPort": 443,
          "protocol": "tcp",
          "appProtocol": "http"
        }
      ],
      "essential": true,
      "environment": [
        {
          "name": "HOST",
          "value": "localhost"
        }
```

```
      ],
      "mountPoints": [],
      "volumesFrom": [],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "true",
          "awslogs-group": "/ecs/graph-explorer",
          "awslogs-region": "{region}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    }
  ],
  "taskRoleArn": "arn:aws:iam::{account_no}:role/{role_name_from_step_1}",
  "executionRoleArn": "arn:aws:iam::{account_no}:role/{role_name_from_step_1}",
  "networkMode": "awsvpc",
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "cpu": "1024",
  "memory": "3072",
  "runtimePlatform": {
    "cpuArchitecture": "X86_64",
    "operatingSystemFamily": "LINUX"
  }
}
```

4. Mulai tugas baru menggunakan pengaturan default, kecuali untuk bidang berikut:

   - Lingkungan
     - Opsi komputasi => Jenis peluncuran
   - Konfigurasi deployment
     - Jenis Aplikasi => Tugas
     - Keluarga => *(your new JSON task definition)*
     - Revisi => *(latest)*
   - Jaringan
     - VPC => *(the Neptune VPC you want to connect to)*
     - Subnet => *(ONLY the public subnets of the VPC– remove all others)*
     - Grup keamanan => Buat grup keamanan baru

- Nama grup keamanan => graph-explorer
- Deskripsi grup keamanan = Grup keamanan untuk akses ke graph-explorer
- Aturan masuk untuk grup keamanan =>
  1. 80 Dimanapun
  2. 443 Dimanapun

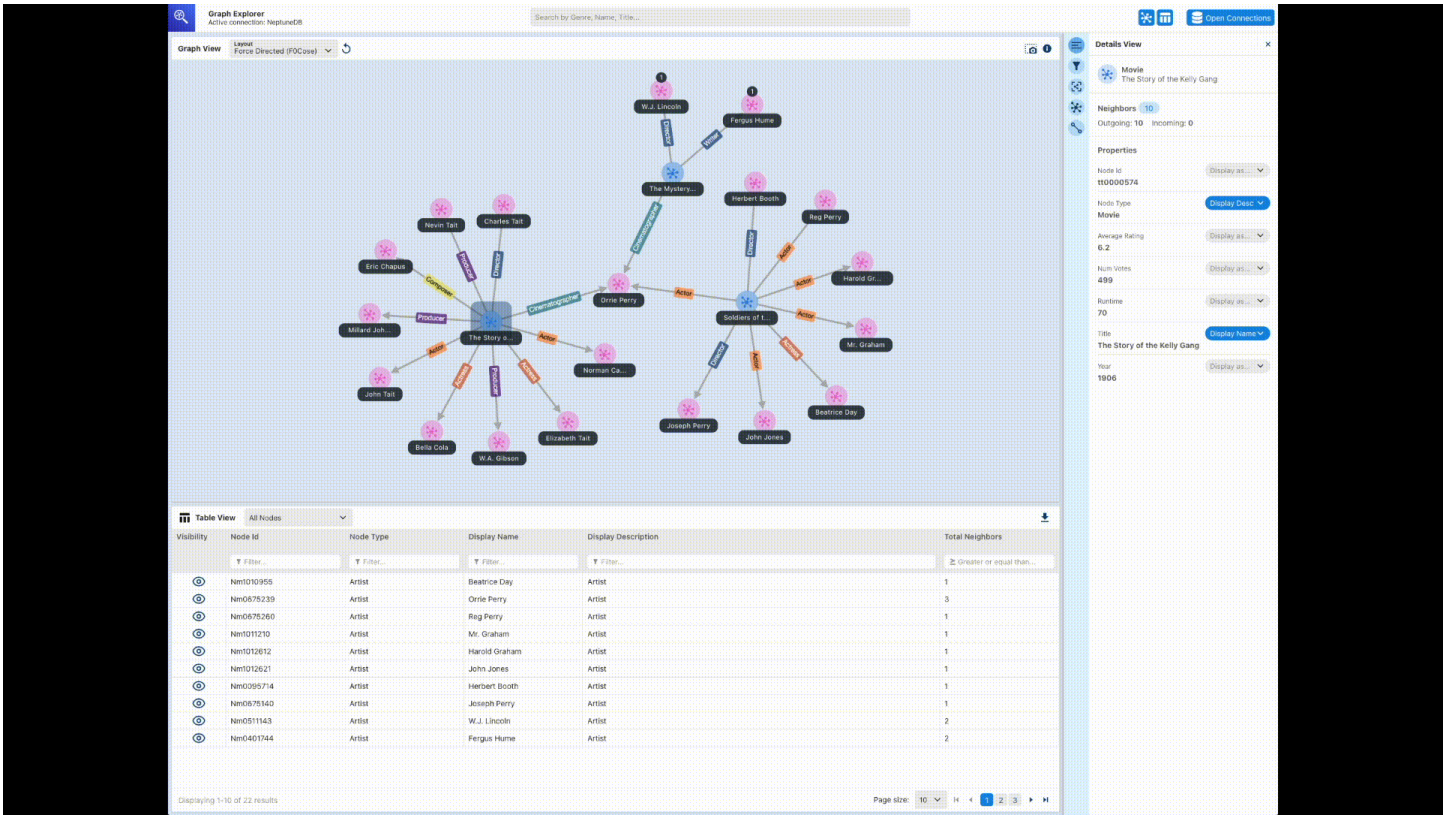
5. Pilih Buat.
6. Setelah tugas dimulai, salin IP publik dari tugas yang sedang berjalan, dan arahkan ke:`https://(your public IP)/explorer`.
7. Terima risiko menggunakan sertifikat yang tidak dikenal yang telah dihasilkan, atau tambahkan ke gantungan kunci Anda.
8. Sekarang Anda dapat menambahkan koneksi ke Neptunus. Buat koneksi baru, baik untuk grafik properti (LPG) atau untuk RDF, dan atur bidang berikut:

```
Using proxy server => true
Public or Proxy Endpoint => https://(your public IP address)
Graph connection URL => https://(your Neptune endpoint):8182
```

Anda sekarang harus terhubung.

Demonstrasi grafik-explorer

Video singkat ini memberi Anda beberapa gambaran tentang bagaimana Anda dapat dengan mudah memvisualisasikan data grafik Anda menggunakan graph-explorer:



Perangkat Lunak Tom Sawyer

Tom Sawyer Perspectives adalah grafik kode rendah dan visualisasi data dan platform pengembangan analisis untuk data yang disimpan di Amazon Neptunus. Antarmuka desain dan pratinjau terintegrasi dan pustaka API yang luas memungkinkan Anda membuat aplikasi visualisasi berkualitas produksi khusus dengan cepat. Dengan antarmuka point-and-click desainer dan 30 algoritma analitik bawaan, Anda dapat merancang dan mengembangkan aplikasi untuk mendapatkan wawasan tentang data yang digabungkan dari lusinan sumber.

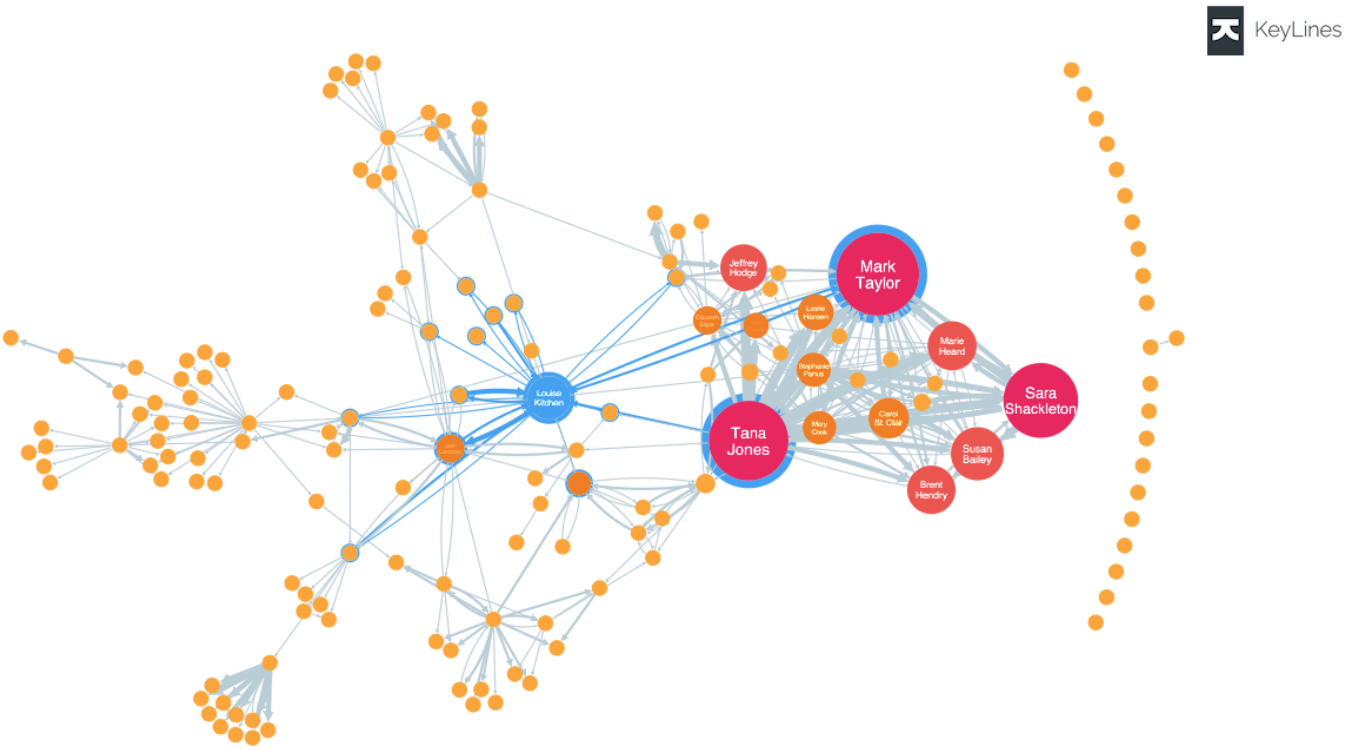
Tom Sawyer Graph Database Browser memudahkan untuk memvisualisasikan dan menganalisis data di Amazon Neptunus. Anda dapat melihat dan memahami koneksi dalam data Anda tanpa pengetahuan yang luas tentang bahasa kueri atau skema. Anda dapat berinteraksi dengan data tanpa pengetahuan teknis hanya dengan memuat tetangga node yang dipilih dan membangun visualisasi ke arah mana pun yang Anda butuhkan. Anda juga dapat memanfaatkan lima tata letak grafik unik untuk menampilkan grafik dengan cara yang memberikan makna paling besar, dan dapat menerapkan analisis sentralitas, pengelompokan, dan pencarian jalur untuk mengungkapkan pola yang sebelumnya tidak terlihat. Untuk melihat contoh integrasi Browser Database Grafik dengan Neptunus, lihat posting blog ini. Untuk memulai uji coba gratis Graph Database Browser, kunjungi AWS Marketplace.

Kecerdasan Cambridge

Cambridge Intelligence menyediakan teknologi visualisasi data untuk mengeksplorasi dan memahami data Amazon Neptunus. Toolkit visualisasi grafik (KeyLinesuntuk JavaScript pengembang dan pengembang React) menawarkan cara mudah ReGraphuntuk membangun alat yang sangat interaktif dan dapat disesuaikan untuk aplikasi web. Toolkit ini memanfaatkan WebGL dan HTML5 Canvas untuk kinerja yang cepat, mereka mendukung fungsi analisis grafik tingkat lanjut, dan menggabungkan fleksibilitas dan skalabilitas dengan arsitektur yang aman dan kuat. Ini SDKs bekerja dengan data Neptunus Gremlin dan RDF.

Lihat tutorial integrasi ini untuk data Gremlin, dataSPARQL, dan arsitektur Neptunus.

Berikut adalah contoh KeyLines visualisasi:

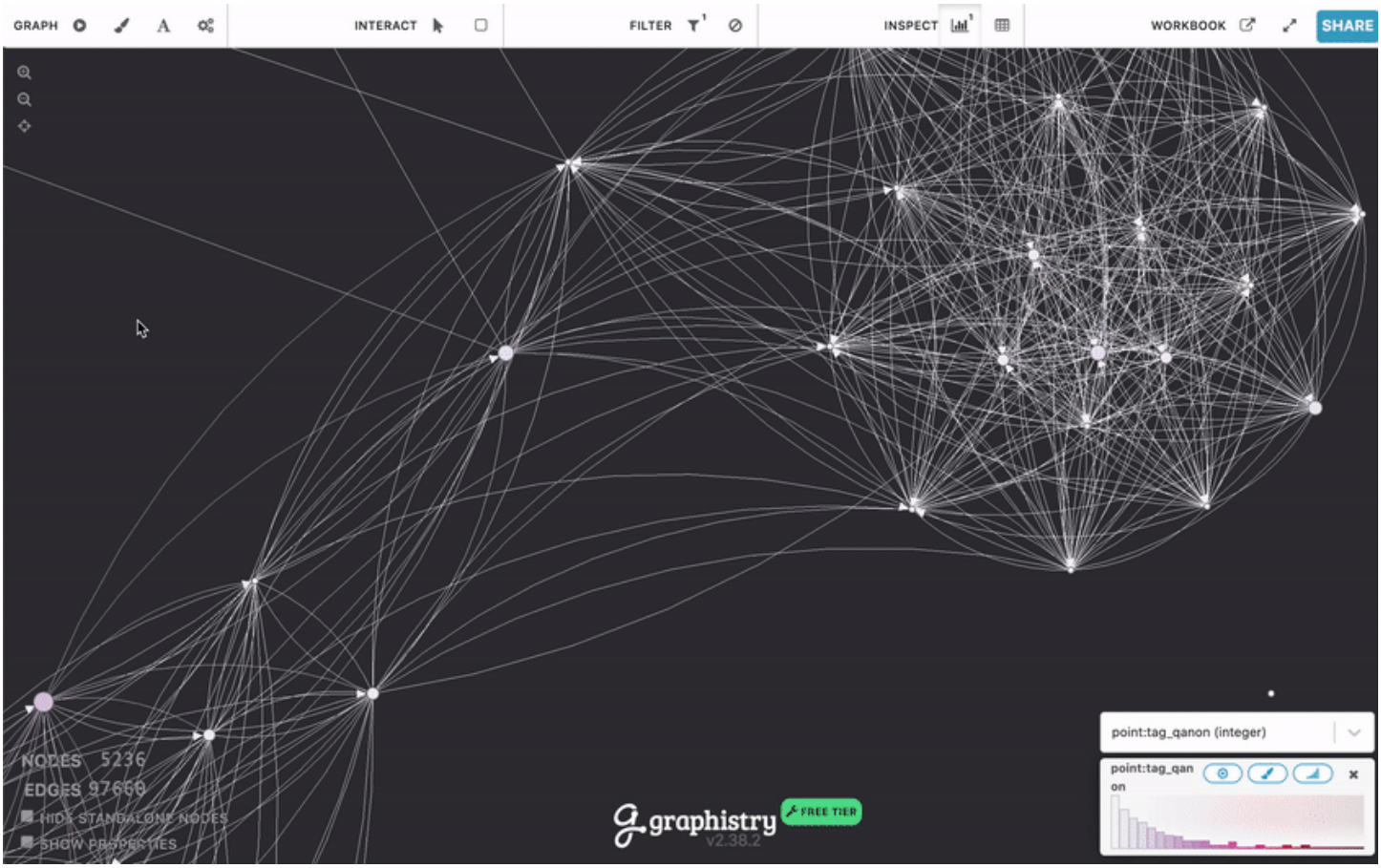


Grafis

Graphistry adalah platform kecerdasan grafik visual yang memanfaatkan akselerasi GPU untuk pengalaman visual yang kaya. Tim dapat berkolaborasi di Graphistry menggunakan berbagai fitur, mulai dari eksplorasi file dan database tanpa kode, hingga berbagi notebook Jupyter dan dasbor Streamlit, hingga menggunakan API penyematan di aplikasi Anda sendiri.

Anda dapat memulai dengan dasbor interaktif penuh pengkodean rendah hanya dengan mengonfigurasi dan meluncurkan graph-app-kitdan memodifikasi hanya beberapa baris kode. Periksa posting blog ini untuk panduan membuat dasbor pertama Anda menggunakan Graphistry dan Neptunus. Anda juga dapat mencoba demo PyGraphistryNeptunus. PyGraphistry adalah pustaka analisis grafik visual Python untuk notebook. Lihat buku catatan tutorial ini untuk demo PyGraphistry Neptunus.

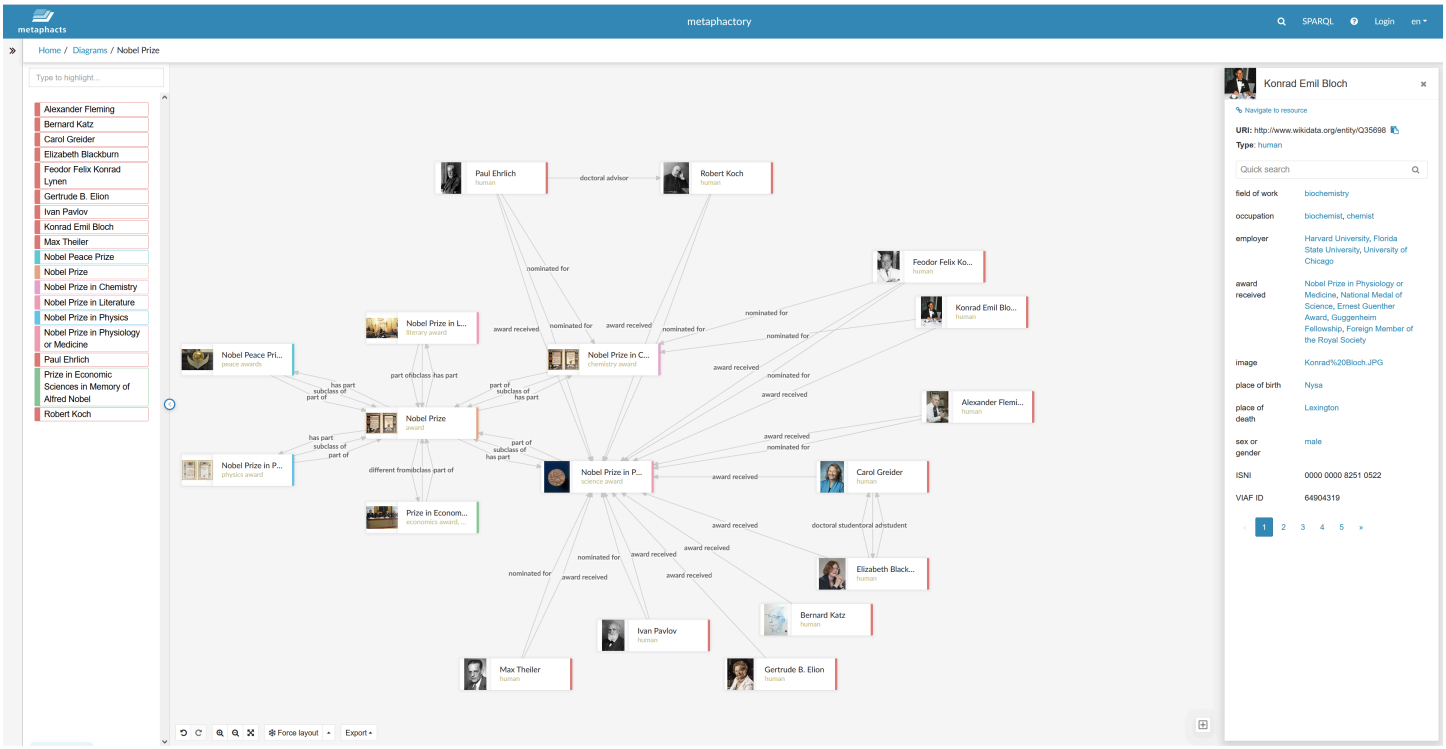
Untuk memulai, kunjungi Graphistry in the Marketplace AWS.



metafak

metafakta menawarkan platform terbuka yang fleksibel untuk menggambarkan dan menanyakan data grafik dan untuk memvisualisasikan dan berinteraksi dengan grafik pengetahuan. Menggunakan metafaktori, Anda dapat membangun aplikasi web interaktif seperti visualisasi dan dasbor di atas grafik pengetahuan di Neptunus menggunakan model data RDF. Platform metafaktori mendukung pengalaman pengembangan kode rendah dengan UI untuk pemuatan data, antarmuka pemodelan ontologi visual dengan dukungan OWL dan SHACL, UI kueri SPARQL dan katalog kueri, dan serangkaian komponen Web yang kaya untuk eksplorasi grafik, visualisasi, pencarian dan penulisan.

Berikut adalah contoh visualisasi metafaktori:



Platform ini dirancang untuk dan digunakan secara produktif di bidang teknik, manufaktur, farmasi, ilmu kehidupan, keuangan, asuransi, dan banyak lagi. Untuk melihat contoh arsitektur solusi, lihat posting blog ini.

Untuk memulai dengan uji coba metafaktori gratis, kunjungi Marketplace.AWS

Klien database grafik G.V ()

G.V () adalah klien database all-in-one grafik yang dibuat untuk pengembang dan profesional data untuk mengeksplorasi dan berinteraksi dengan data grafik Amazon Neptunus.

Dengan G.V (), Anda dapat mempercepat proyek grafik dan melakukan tugas database day-to-day grafik dengan lebih efisien. Ini tersedia melalui AWS Marketplace atau sebagai aplikasi desktop mandiri untuk Windows, macOS, dan Linux.

Fitur utama

Editor kueri

Editor kueri Gremlin dan Cypher bawaan menyediakan pelengkapan otomatis, saran, dan dokumentasi waktu nyata saat Anda mengetik kueri. Fitur-fitur ini diinformasikan oleh skema data grafik Anda untuk membantu Anda menulis kueri yang akurat dengan cepat.

Anda dapat membuat parameter dan menyimpan kueri sebagai laporan sekali klik, memungkinkan Anda membuat pelaporan terpusat dan berwawasan luas untuk database grafik Anda dalam hitungan menit.

Format visualisasi data

Untuk menganalisis hasil kueri Anda, G.V () menawarkan beberapa format visualisasi data:

- Visualisasi grafik untuk menjelajahi koneksi dan pola dalam data Anda, dengan dasar opsional untuk data peta geografis
- Tampilan tabel, mengatur simpul, tepi, atau hasil kueri tabel sebagai baris untuk perbandingan cepat nilai properti
- Format data JSON untuk output yang ramah pengembang
- Objek browser untuk menavigasi data Anda berdasarkan struktur hierarkisnya, paling cocok untuk agregasi informasi yang kompleks

Penjelajah data grafik

Graph Data Explorer tanpa kode G.V () memungkinkan Anda mencari data grafik Anda dengan mudah menggunakan filter properti dan melintasi tepi dari simpul secara interaktif melalui tampilan grafik. Anda juga dapat membuat, memperbarui, dan menghapus simpul, tepi, dan propertinya dengan beberapa klik untuk memodifikasi data grafik Anda tanpa menulis kueri lengkap.

Tampilan model data grafik

Tampilan Model Data Grafik memberikan representasi visual terperinci dari skema grafik properti berlabel Anda untuk membantu tim Anda memahami model data Anda.

Integrasi Amazon Neptunus

G.V () menawarkan integrasi mendalam dengan Amazon Neptunus, termasuk:

- Autentikasi IAM
- Dukungan Neptunus API
- Analisis Kueri Lambat
- Wawasan Audit Log

Lihat dokumentasi G.V () untuk mempelajari lebih lanjut.

Ketersediaan dan pembaruan

G.V () terus berkembang dengan fitur-fitur baru yang dirilis setiap bulan. Mulailah dengan uji coba gratis versi desktop atau AWS Marketplace hari ini.

Demonstrasi produk

Lihat G.V () beraksi:

Pelajari selengkapnya

Untuk informasi selengkapnya tentang fitur dan kemampuan G.V (), lihat dokumentasi G.V ().

Linkurious

Linkurious menyediakan solusi intelijen grafik yang berbeda untuk pengguna teknis dan non-teknis dan berbagai kasus penggunaan.

Linkurious Enterprise Explorer adalah perangkat lunak visualisasi dan analisis off-the-shelf grafik yang dibuat untuk tim yang dapat memenuhi tuntutan day-to-day aktivitas Anda dan membantu para profesional berbasis data melakukan hal-hal besar - sederhana. Sepenuhnya dapat dikonfigurasi dan mudah digunakan, dengan mudah menyesuaikan dengan kebutuhan Anda dan memberdayakan pemula atau pengguna tingkat lanjut untuk memvisualisasikan data dengan cepat di AWS Neptunus, untuk menjelajahi kumpulan data Anda secara intuitif terlepas dari ukuran atau kompleksitas data Anda dan untuk berkolaborasi dengan mulus di tingkat tim atau perusahaan.

Linkurious Enterprise Watchtower memanfaatkan kekuatan Linkurious Enterprise Explorer dan menambahkan kemampuan deteksi dan manajemen kasus yang inovatif untuk menawarkan perangkat lunak deteksi dan investigasi terintegrasi yang didukung oleh teknologi grafik. Di satu sisi, ini memungkinkan Anda mengonfigurasi peringatan yang memanfaatkan Database Neptunus dan Neptunus Analytics untuk secara otomatis memunculkan anomali atau pola dalam data terhubung yang kompleks. Di sisi lain, ini menggabungkan fitur manajemen kasus dan kolaborasi untuk membantu tim mengelola alur kerja investigasi mereka secara efisien.

Ogma adalah JavaScript perpustakaan komersial yang membantu Anda mengembangkan visualisasi grafik interaktif skala besar yang kuat untuk aplikasi Anda. Ini memanfaatkan rendering WebGL dan tata letak kinerja tinggi untuk memungkinkan pengguna menampilkan dan berinteraksi dengan ribuan node dan tepi dalam hitungan detik. Ini juga menyediakan berbagai fitur untuk menyesuaikan

aplikasi Anda dan menciptakan pengalaman pengguna yang kaya. Akhirnya, ia dilengkapi dengan dokumentasi dan alat yang komprehensif seperti tutorial, lusinan contoh dan taman bermain interaktif.

Untuk memulai, mintalah uji coba gratis 30 hari Linkurious Enterprise atau Ogma.

Graph.Build

Dalam domain apa pun, kolaborasi dengan pakar materi pelajaran adalah kunci untuk merancang model grafik yang secara efektif menangani kasus penggunaan tertentu. Baik Anda menangani beban kerja grafik tradisional seperti analitik grafik atau menjelajahi aplikasi AI tingkat lanjut seperti GraphRag, prediksi tautan, atau penalaran neuro-simbolik, kemampuan untuk membangun dan mengulangi model grafik sangat penting. Dengan Graph.Build, merancang dan menyempurnakan model grafik menjadi lebih mudah diakses, memberdayakan pakar domain untuk bekerja hand-in-hand dengan insinyur dan analis data untuk menciptakan solusi yang berdampak dan disesuaikan.

Platform Graph.Build adalah studio desain model grafik dan pabrik pembuatan khusus tanpa kode.

Graph.Build tersedia di Marketplace AWS .

Graph.Build Studio:

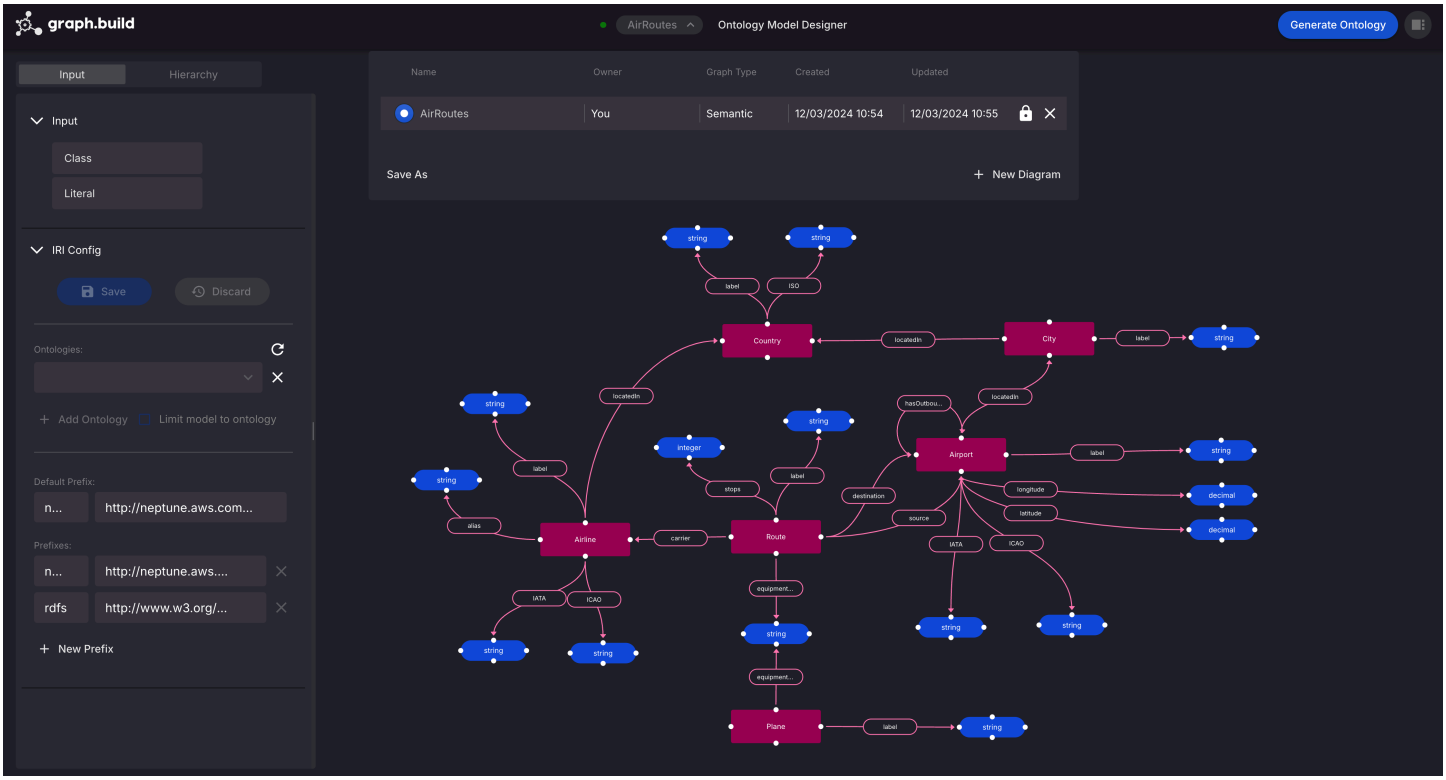
Studio adalah IDE desain grafik kolaboratif tanpa kode. Anda dapat menggunakan studio untuk merancang skema untuk grafik LPG, Ontologi untuk grafik RDF, model ETL, Ubah konfigurasi Pengambilan Data, dan bahkan titik akhir grafik SPARQL virtual.

Graf.Build Transformer:

Transformer membuat model grafik dari data nyata. Model grafik transformator dapat dirancang di Studio, artinya merancang, membangun, dan menguji model grafik tidak memerlukan kode. Anda dapat menghubungkan transformer ke berbagai sumber data termasuk Amazon Aurora, Amazon Athena, AmazonS3, dan Amazon MSK. Transformer memiliki berbagai fitur untuk mempertahankan model grafik yang ditemukan di kumpulan data lain, termasuk perubahan pengambilan data (CDC), virtualisasi dan materialisasi dalam mode pembaruan atau penyisipan.

Penulis Graph.Build:

Penulis dikonfigurasi untuk menulis atau memperbarui model grafik Grafik Properti Berlabel (LPG) dan Kerangka Deskripsi Sumber Daya (RDF) dalam basis data grafik target, seperti Amazon Neptunus dan Neptunus Analytics.



[Minta demo](#) hari ini.

Mengekspor data dari cluster DB Neptunus

Ada beberapa cara bagus untuk mengekspor data dari cluster DB Neptunus:

- Untuk sejumlah kecil data, cukup gunakan hasil kueri atau kueri.
- Untuk data RDF, Graph Store Protocol (GSP) dapat mempermudah ekspor. Contoh:

```
curl --request GET \
  'https://your-neptune-endpoint:port/sparql/gsp/?graph=http://www.example.com/named/
graph'
```

- Ada juga alat open-source yang kuat dan fleksibel untuk mengekspor data Neptunus, yaitu. `neptune-export` Bagian berikut menjelaskan fitur alat ini dan cara menggunakannya.

Topik

- Menggunakan neptune-export
- Menggunakan layanan Neptunus-Ekspor untuk mengekspor data Neptunus
- Menggunakan alat neptune-export baris perintah untuk mengekspor data dari Neptunus
- File yang diekspor oleh Neptune-Ekspor dan neptune-export
- Parameter yang digunakan untuk mengontrol proses ekspor Neptunus
- Memecahkan masalah proses ekspor Neptunus
- Mengekspor hasil kueri Gremlin ke Amazon S3

Menggunakan **`neptune-export`**

Anda dapat menggunakan `neptune-export`alat sumber terbuka dengan dua cara berbeda:

- Sebagai layanan Neptunus-Ekspor. Saat Anda mengekspor data dari Neptunus menggunakan layanan Neptunus-Ekspor, Anda memicu dan memantau pekerjaan ekspor melalui REST API.
- Sebagai utilitas baris perintah **`neptune-export`** Java. Untuk menggunakan alat baris perintah ini untuk mengekspor data Neptunus, Anda harus menjalankannya di lingkungan di mana cluster DB Neptunus Anda dapat diakses.

Baik layanan Neptunus-Ekspor dan alat baris `neptune-export` perintah mempublikasikan data ke Amazon Simple Storage Service (Amazon S3), dienkripsi menggunakan enkripsi sisi server Amazon S3 (). `SSE-S3`

> **Note**
>
> Ini adalah praktik terbaik untuk mengaktifkan pencatatan akses di semua bucket Amazon S3, untuk memungkinkan Anda mengaudit semua akses ke bucket tersebut.

Jika Anda mencoba untuk mengekspor data dari klaster DB Neptune yang datanya berubah saat ekspor terjadi, konsistensi data yang diekspor tidak dijamin. Artinya, jika klaster Anda melayani lalu lintas tulis sementara pekerjaan ekspor sedang berlangsung, mungkin ada inkonsistensi dalam data yang diekspor. Hal ini berlaku apakah Anda mengekspor dari instans utama dalam klaster atau dari satu replika baca atau lebih.

Untuk menjamin bahwa data yang diekspor konsisten, sebaiknya mengekspor dari klon klaster DB Anda. Ini menyediakan alat ekspor dengan versi statis data Anda dan memastikan bahwa pekerjaan ekspor tidak memperlambat kueri di cluster DB asli Anda.

Untuk membuatnya lebih mudah, Anda dapat menunjukkan bahwa Anda ingin mengkloning klaster DB sumber ketika Anda memicu pekerjaan ekspor. Jika Anda melakukannya, proses ekspor secara otomatis membuat klon, menggunakannya untuk ekspor, dan kemudian menghapusnya ketika ekspor selesai.

Menggunakan layanan Neptunus-Ekspor untuk mengekspor data Neptunus

Anda dapat menggunakan langkah-langkah berikut untuk mengekspor data dari klaster DB Neptune Anda ke Amazon S3 menggunakan layanan Neptune-Ekspor:

Menginstal layanan Neptunus-Ekspor

Gunakan AWS CloudFormation template untuk membuat tumpukan:

Menginstal layanan Neptune-Ekspor

1. Luncurkan CloudFormation tumpukan di CloudFormation konsol dengan memilih salah satu tombol Launch Stack di tabel berikut:

Wilayah	Lihat	Lihat di Designer	Luncurkan
US East (N. Virginia)	Lihat	Lihat di Desainer	
AS Timur (Ohio)	Lihat	Lihat di Desainer	
US West (N. California)	Lihat	Lihat di Desainer	
US West (Oregon)	Lihat	Lihat di Desainer	
Kanada (Pusat)	Lihat	Lihat di Desainer	
Amerika Selatan (Sao Paulo)	Lihat	Lihat di Desainer	
Europe (Stockholm)	Lihat	Lihat di Desainer	
Eropa (Irlandia)	Lihat	Lihat di Desainer	
Eropa (London)	Lihat	Lihat di Desainer	
Europe (Paris)	Lihat	Lihat di Desainer	
Eropa (Frankfurt)	Lihat	Lihat di Desainer	
Timur Tengah (Bahrain)	Lihat	Lihat di Desainer	
Timur Tengah (UEA)	Lihat	Lihat di Desainer	
Israel (Tel Aviv)	Lihat	Lihat di Desainer	

Wilayah	Lihat	Lihat di Designer	Luncurkan
Afrika (Cape Town)	Lihat	Lihat di Desainer	Launch Stack
Asia Pasifik (Hong Kong)	Lihat	Lihat di Desainer	Launch Stack
Asia Pacific (Tokyo)	Lihat	Lihat di Desainer	Launch Stack
Asia Pasifik (Seoul)	Lihat	Lihat di Desainer	Launch Stack
Asia Pacific (Singapore)	Lihat	Lihat di Desainer	Launch Stack
Asia Pacific (Sydney)	Lihat	Lihat di Desainer	Launch Stack
Asia Pasifik (Mumbai)	Lihat	Lihat di Desainer	Launch Stack
Tiongkok (Beijing)	Lihat	Lihat di Desainer	Launch Stack
Tiongkok (Ningxia)	Lihat	Lihat di Desainer	Launch Stack
AWS GovCloud (AS-Barat)	Lihat	Lihat di Desainer	Launch Stack
AWS GovCloud (AS-Timur)	Lihat	Lihat di Desainer	Launch Stack

2. Pada halaman Pilih Templat, pilih Selanjutnya.
3. Pada halamanTentukan Detail, template, mengatur parameter berikut:

   - **VPC** – Cara termudah untuk mengatur layanan Neptune-Ekspor adalah dengan menginstalnya di Amazon VPC yang sama dengan database Neptune Anda. Jika Anda ingin menginstalnya di VPC terpisah Anda bisa gunakanPeering VPC untuk membangun konektivitas antara VPC klaster DB Neptune dan VPC layanan Neptune-Ekspor.

- **`Subnet1`**— Layanan Neptunus-Ekspor harus dipasang di subnet di VPC Anda yang memungkinkan lalu lintas HTTPS IPv4 keluar dari subnet ke internet. Hal ini agar layanan Neptune-Ekspor dapat menghubungi AWS Batch API untuk membuat dan menjalankan pekerjaan ekspor.

  Jika Anda membuat cluster Neptunus menggunakan CloudFormation template pada Buat cluster Neptunus halaman dalam dokumentasi Neptunus, Anda dapat menggunakan `PrivateSubnet2` dan output dari tumpukan `PrivateSubnet1` itu untuk mengisi parameter ini dan parameter berikutnya.
- **`Subnet2`**Subnet kedua di VPC yang memungkinkan lalu lintas HTTPS IPv4 keluar dari subnet ke internet.
- **`EnableIAM`** – Atur ini ke `true` untuk mengamankan API Neptune-Titik akhir menggunakan (IAM) AWS Identity and Access Management . Kami sarankan Anda melakukan juga.

  Jika Anda mengaktifkan autentikasi IAM, Anda harus `Sigv4` menandatangani semua permintaan HTTPS ke titik akhir. Anda dapat menggunakan alat seperti awscurl untuk menandatangani permintaan atas nama Anda.
- **`VPCOnly`** – Mengatur ini ke `true` menyebabkan ekspor titik akhir VPC-saja, sehingga Anda hanya dapat mengaksesnya dari dalam VPC tempat layanan Neptune-ekspor diinstal. Ini membatasi API Neptune-Ekspor untuk digunakan hanya dari dalam VPC itu.

  Kami sarankan Anda mengatur `VPCOnly` ke `true`.
- **`NumOfFilesULimit`** — Tentukan nilai antara 10.000 dan 1.000.000 untuk `nofile` di properti `ulimits` kontainer. Defaultnya adalah 10.000, dan kami sarankan untuk menjaga default kecuali grafik Anda berisi sejumlah besar label unik.
- **`PrivateDnsEnabled`** (Boolean) - Menunjukkan apakah akan mengaitkan zona host pribadi dengan VPC yang ditentukan atau tidak. Nilai default-nya adalah `true`.

  Saat titik akhir VPC dibuat dengan flag ini diaktifkan, semua lalu lintas API Gateway dirutekan melalui titik akhir VPC, dan panggilan titik akhir API Gateway publik menjadi dinonaktifkan. Jika disetel `PrivateDnsEnabled` kefalse, titik akhir API Gateway publik diaktifkan, tetapi layanan ekspor Neptunus tidak dapat dihubungkan melalui titik akhir DNS pribadi. Anda kemudian dapat menggunakan titik akhir DNS publik untuk titik akhir VPC untuk memanggil layanan ekspor, seperti yang dijelaskan di sini.
- **`NeptuneExportVersion`**— Tentukan versi Utilitas Ekspor Neptunus yang akan digunakan. Semua versi yang lebih besar dari atau sama dengan `v1.1.11` didukung. Versi `v2.latest`

dapat digunakan untuk secara otomatis menerima pembaruan kecil. Daftar lengkap versi yang tersedia, serta catatan tambalan dapat ditemukan di GitHub rilis open source.

4. Pilih Berikutnya.
5. Pada halaman Opsi, pilih Selanjutnya.
6. Pada halaman Review, pilih kotak centang pertama untuk mengetahui bahwa CloudFormation akan membuat sumber daya IAM. Pilih kotak centang kedua untuk mengetahui `CAPABILITY_AUTO_EXPAND` untuk tumpukan baru.

> **Note**
>
> `CAPABILITY_AUTO_EXPAND` secara eksplisit mengakui bahwa macro akan diperluas saat membuat tumpukan, tanpa review sebelumnya. Pengguna sering kali membuat perubahan yang ditetapkan dari templat yang diproses, sehingga perubahan yang dibuat oleh makro bisa direview tepat sebelum membuat tumpukan. Untuk informasi selengkapnya, lihat CloudFormation CreateStackAPI.

Lalu pilih Buat.

Aktifkan akses ke Neptune dari Neptune-Ekspor

Setelah instalasi Neptune-Ekspor selesai, perbarui Grup keamanan VPC Neptune Anda untuk mengizinkan akses dari Neptune-Ekspor. Ketika CloudFormation tumpukan Neptunus-Ekspor telah dibuat, tab Output menyertakan ID. `NeptuneExportSecurityGroup` Perbarui grup keamanan Neptune VPC Anda untuk mengizinkan akses dari grup keamanan Neptune-Ekspor ini.

Aktifkan akses ke titik akhir Neptunus-Ekspor dari instance berbasis VPC EC2

Jika Anda membuat titik akhir Neptune-Ekspor VPC saja, Anda hanya dapat mengaksesnya dari dalam VPC tempat layanan Neptune-Ekspor diinstal. Untuk mengizinkan konektivitas dari EC2 instans Amazon di VPC tempat Anda dapat melakukan panggilan Neptune-Export API, lampirkan yang `NeptuneExportSecurityGroup` dibuat oleh tumpukan CloudFormation ke instance Amazon tersebut. EC2

Jalankan tugas Neptune-Ekspor menggunakan API Neptune-Ekspor

Tab Output dari CloudFormation tumpukan juga menyertakan file. `NeptuneExportApiUri` Gunakan URI ini setiap kali Anda mengirim permintaan ke titik akhir Neptune-Ekspor.

Menjalankan tugas ekspor

- Pastikan bahwa pengguna atau peran di mana ekspor berjalan telah diberikan Izin `execute-api:Invoke`.
- Jika Anda mengatur parameter `EnableIAM` ke `true` di tumpukan CloudFormation ketika Anda menginstal Neptune-Ekspor, Anda harus `Sigv4` tandatangani semua permintaan ke API Neptune-Ekspor. Kami menyarankan penggunaan awscurl untuk membuat permintaan ke API. Semua contoh di sini menganggap bahwa otentikasi IAM diaktifkan.
- Jika Anda menyetel `VPCOnly` parameter ke `true` dalam CloudFormation tumpukan saat menginstal Neptune-Export, Anda harus memanggil Neptune-Export API dari dalam VPC, biasanya dari instance Amazon yang terletak di VPC. EC2

Untuk mulai mengekspor data, kirim permintaan ke `NeptuneExportApiUri` titik akhir dengan parameter `outputS3Path` permintaan `command` dan parameter `endpoint` ekspor.

Berikut ini adalah contoh permintaan yang mengekspor data grafik properti dari Neptunus dan menerbitkannya ke Amazon S3:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-pg",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": { "endpoint": "(your Neptune endpoint DNS name)" }
      }'
```

Demikian pula, berikut adalah contoh permintaan yang mengekspor data RDF dari Neptunus ke Amazon S3:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
```

```
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-rdf",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": { "endpoint": "(your Neptune endpoint DNS name)" }
      }'
```

Jika Anda menghilangkan parameter `command` permintaan, secara default Neptunus-Ekspor mencoba mengekspor data grafik properti dari Neptunus.

Jika perintah sebelumnya berhasil, output akan terlihat seperti ini:

```
{
  "jobName": "neptune-export-abc12345-1589808577790",
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f"
}
```

Pantau tugas ekspor yang baru saja Anda mulai

Untuk memantau tugas yang sedang berjalan, tambahkan JobID ke `NeptuneExportApiUri` Anda, sesuatu seperti ini:

```
curl \
  (your NeptuneExportApiUri)/(the job ID)
```

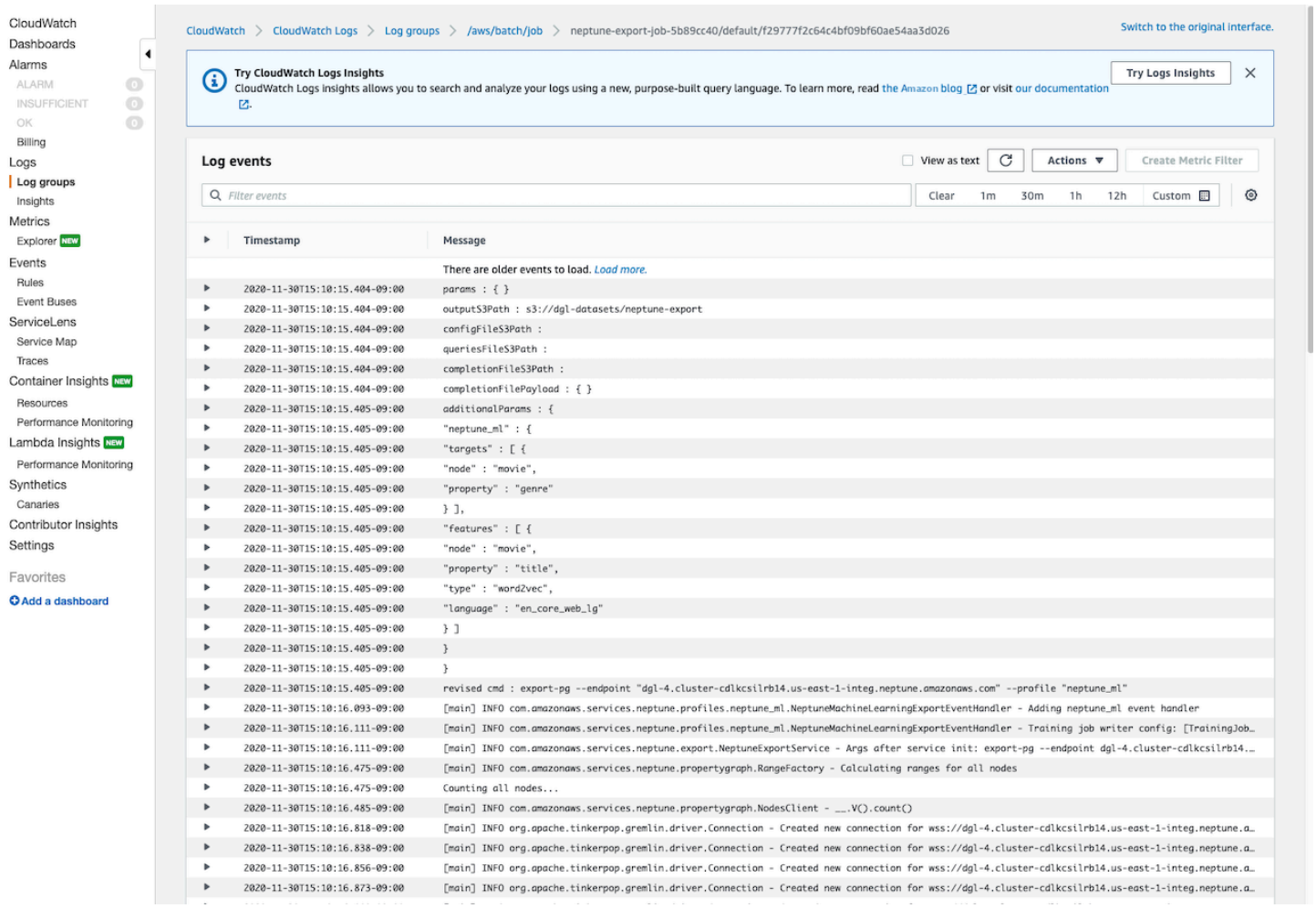
Jika layanan belum memulai tugas ekspor, responnya akan terlihat seperti ini:

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "pending"
}
```

Ketika Anda mengulangi perintah setelah tugas ekspor dimulai, responnya akan terlihat seperti ini:

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "running",
  "logs": "https://us-east-1.console.aws.amazon.com/cloudwatch/home?..."
}
```

Jika Anda membuka CloudWatch log di Log menggunakan URI yang disediakan oleh panggilan status, Anda kemudian dapat memantau kemajuan ekspor secara detail:



Membatalkan pekerjaan ekspor yang sedang berjalan

Untuk membatalkan pekerjaan ekspor yang sedang berjalan menggunakan Konsol Manajemen AWS

1. Buka AWS Batch konsol di https://console.aws.amazon.com/batch/.
2. Pilih Pekerjaan.
3. Temukan pekerjaan yang sedang berjalan yang ingin Anda batalkan, berdasarkan tugasnya`jobID`.
4. Pilih Batalkan pekerjaan.

Untuk membatalkan pekerjaan ekspor yang sedang berjalan menggunakan API ekspor Neptunus:

Kirim `HTTP DELETE` permintaan ke `NeptuneExportApiUri` dengan yang `jobID` ditambahkan, seperti ini:

```
curl -X DELETE \
  (your NeptuneExportApiUri)/(the job ID)
```

Menggunakan alat **`neptune-export`** baris perintah untuk mengekspor data dari Neptunus

Anda dapat menggunakan langkah-langkah berikut untuk mengekspor data dari cluster DB Neptunus ke Amazon S3 menggunakan utilitas baris perintah: `neptune-export`

Prasyarat untuk menggunakan utilitas baris perintah **`neptune-export`**

Sebelum Anda mulai

- Memiliki versi 8 dari JDK — Anda memerlukan versi 8 Java SE Development Kit (JDK) diinstal.
- Unduh utilitas neptune-ekspor – Unduh dan instal file neptune-export.jar
- Pastikan **`neptune-export`** memiliki akses ke VPC Neptune Anda – Jalankan neptune-ekspor dari lokasi tempatnya dapat mengakses VPC di mana klaster DB Neptune Anda berada.

  Misalnya, Anda dapat menjalankannya pada instans Amazon EC2 dalam VPC Neptune, atau di VPC terpisah yang dipasangkan dengan Neptune VPC, atau pada host bastion terpisah.
- Pastikan grup keamanan VPC memberikan akses ke **`neptune-export`** – Periksa apakah grup keamanan VPC yang terpasang pada VPC Neptune mengizinkan akses ke klaster DB Anda dari alamat IP atau grup keamanan yang terkait dengan lingkungan `neptune-export`.
- Siapkan izin IAM yang diperlukan — Jika database Anda mengaktifkan autentikasi database AWS Identity and Access Management (IAM), pastikan peran yang `neptune-export` dijalankan dikaitkan dengan kebijakan IAM yang memungkinkan koneksi ke Neptunus. Untuk informasi tentang kebijakan Neptune, lihat Menggunakan kebijakan IAM.

  Jika Anda ingin menggunakan parameter ekspor `clusterId` dalam permintaan kueri Anda, peran tempat `neptune-export` berjalan memerlukan izin IAM berikut:
  - `rds:DescribeDBClusters`
  - `rds:DescribeDBInstances`
  - `rds:ListTagsForResource`

Jika Anda ingin mengekspor dari cluster kloning, peran tempat `neptune-export` berjalan memerlukan izin IAM berikut:

- `rds:AddTagsToResource`
- `rds:DescribeDBClusters`
- `rds:DescribeDBInstances`
- `rds:ListTagsForResource`
- `rds:DescribeDBClusterParameters`
- `rds:DescribeDBParameters`
- `rds:ModifyDBParameterGroup`
- `rds:ModifyDBClusterParameterGroup`
- `rds:RestoreDBClusterToPointInTime`
- `rds:DeleteDBInstance`
- `rds:DeleteDBClusterParameterGroup`
- `rds:DeleteDBParameterGroup`
- `rds:DeleteDBCluster`
- `rds:CreateDBInstance`
- `rds:CreateDBClusterParameterGroup`
- `rds:CreateDBParameterGroup`

Untuk mempublikasikan data yang diekspor ke Amazon S3, peran tempat `neptune-export` berjalan memerlukan izin IAM berikut untuk lokasi Amazon S3:

- `s3:PutObject`
- `s3:PutObjectTagging`
- `s3:GetObject`

- Mengatur variabel lingkungan**`SERVICE_REGION`** — Mengatur variabel lingkungan `SERVICE_REGION` untuk mengidentifikasi Wilayah letak cluster DB Anda (lihat Menghubungkan ke Neptuneuntuk daftar pengidentifikasi Wilayah).

Menjalankan utilitas **`neptune-export`** untuk memulai operasi ekspor

Gunakan perintah berikut untuk menjalankan neptune-ekspor dari baris perintah dan memulai operasi ekspor:

```
java -jar neptune-export.jar nesvc \
  --root-path (path to a local directory) \
  --json (the JSON file that defines the export)
```

Perintah ini memiliki dua parameter:

Parameter untuk neptune-ekspor ketika memulai ekspor

- **`--root-path`** – Path ke direktori lokal tempat file ekspor ditulis sebelum dipublikasikan ke Amazon S3.
- **`--json`** – Sebuah objek JSON yang mendefinisikan ekspor.

Contoh perintah menggunakan utilitas baris perintah **`neptune-export`**

Untuk mengekspor data grafik properti langsung dari cluster DB sumber Anda:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
            }
          }'
```

Untuk mengekspor data RDF langsung dari cluster DB sumber Anda:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-rdf",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
            }
          }'
```

Jika Anda menghilangkan parameter `command` permintaan, `neptune-export` utilitas mengekspor data grafik properti dari Neptunus secara default.

Untuk mengekspor dari klon cluster DB Anda:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)",
              "cloneCluster" : true
            }
          }'
```

Untuk mengekspor dari cluster DB Anda menggunakan autentikasi IAM:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
              "useIamAuth" : true
            }
          }'
```

File yang diekspor oleh Neptune-Ekspor dan **`neptune-export`**

Ketika ekspor selesai, file ekspor diterbitkan ke lokasi Amazon S3 yang telah Anda tentukan. Semua file yang dipublikasikan ke Amazon S3 dienkripsi menggunakan enkripsi sisi server Amazon S3 (`SSE-S3`). Folder dan file yang dipublikasikan ke Amazon S3 bervariasi tergantung pada apakah Anda mengekspor grafik properti atau data RDF. Jika Anda membuka lokasi Amazon Amazon S3 tempat file diterbitkan, Anda melihat konten berikut:

Lokasi file yang diekspor di Amazon S3

- **`nodes/`**— Folder ini berisi file data node baik dalam nilai dipisahkan koma (CSV) atau format JSON.

  Di Neptune, simpul dapat memiliki satu label atau lebih. Simpul dengan label individu yang berbeda (atau kombinasi yang berbeda dari beberapa label) ditulis ke file yang berbeda, yang berarti bahwa

tidak ada file individual yang berisi data untuk simpul dengan kombinasi label yang berbeda. Jika simpul memiliki beberapa label, label ini diurutkan menurut abjad sebelum ditugaskan ke file.

- **`edges/`**— Folder ini berisi file data tepi baik dalam nilai dipisahkan koma (CSV) atau format JSON.

  Seperti dengan file simpul, data edge ditulis ke file yang berbeda berdasarkan kombinasi label mereka. Untuk tujuan pelatihan model, data edge ditugaskan ke file yang berbeda berdasarkan kombinasi label edge ditambah label dari awal edge dan akhir simpul.
- **`statements/`**— Folder ini berisi file data RDF dalam format Turtle, N-Quads, N-Triples, atau JSON.
- **`config.json`** – File ini berisi Skema dari grafik seperti disimpulkan oleh proses ekspor.
- **`lastEventId.json`**— File ini berisi `commitNum` dan `opNum` peristiwa terakhir pada aliran Neptunus database. Proses ekspor hanya menyertakan file ini jika Anda mengatur parameter `includeLastEventId` ekspor`true`, dan database tempat Anda mengekspor data mengaktifkan aliran Neptunus.

Parameter yang digunakan untuk mengontrol proses ekspor Neptunus

Apakah Anda menggunakan layanan Neptune-Export atau utilitas baris perintah `neptune-export`, parameter yang Anda gunakan untuk mengontrol ekspor sebagian besar sama. Mereka berisi objek JSON diteruskan ke endpoint Neptune-Export atau ke baris perintah `neptune-export`.

Objek yang diteruskan ke proses ekspor memiliki hingga lima bidang tingkat atas:

```
-d '{
      "command" : "(either export-pg or export-rdf)",
      "outputS3Path" : "s3:/(your Amazon S3 bucket)/(path to the folder for exported
 data)",
      "jobSize" : "(for Neptune-Export service only)",
      "params" : { (a JSON object that contains export-process parameters) },
      "additionalParams": { (a JSON object that contains parameters for training
 configuration) }
    }'
```

Daftar Isi

- commandParameternya

- outputS3PathParameternya
- jobSizeParameternya
- paramsObjeknya
- additionalParamsObjeknya
- Ekspor bidang parameter di objek params JSON tingkat atas
  - Daftar bidang yang mungkin di params objek parameter ekspor
    - Daftar bidang yang umum untuk semua jenis ekspor
    - Daftar bidang untuk ekspor grafik properti
    - Daftar bidang untuk ekspor RDF
  - Bidang umum untuk semua jenis ekspor
    - cloneClusterbidang di params
    - cloneClusterInstanceTypebidang di params
    - cloneClusterReplicaCountbidang di params
    - cloneClusterEnableAuditLogsbidang di params
    - clusterIdbidang di params
    - endpointbidang di params
    - endpointsbidang di params
    - profilebidang di params
    - useIamAuthbidang di params
    - includeLastEventIdbidang di params
  - Bidang untuk ekspor grafik properti
    - concurrencybidang di params
    - edgeLabelsbidang di params
    - filterbidang di params
    - filterConfigFilebidang di params
    - formatbidang yang digunakan untuk data grafik properti di params
    - gremlinFilterbidang di params
    - gremlinNodeFilterbidang di params
    - gremlinEdgeFilterbidang di params
    - nodeLabelsbidang di params

    - scopebidang di params
  - Bidang untuk ekspor RDF
    - formatbidang yang digunakan untuk data RDF di params
    - rdfExportScopebidang di params
    - sparqlbidang di params
    - namedGraphbidang di params
- Contoh penyaringan apa yang diekspor
  - Memfilter ekspor data grafik properti
    - Contoh penggunaan scope untuk mengekspor hanya tepi
    - Contoh menggunakan nodeLabels dan edgeLabels mengekspor hanya node dan tepi yang memiliki label tertentu
    - Contoh penggunaan filter untuk mengekspor hanya node, tepi, dan properti tertentu
    - Contoh yang menggunakan gremlinFilter
    - Contoh yang menggunakan gremlinNodeFilter
    - Contoh yang menggunakan gremlinEdgeFilter
    - Menggabungkanfilter,gremlinNodeFilter,nodeLabels, edgeLabels dan scope
  - Memfilter ekspor data RDF
    - Menggunakan rdfExportScope dan sparql mengekspor tepi tertentu
    - Menggunakan namedGraph untuk mengekspor satu grafik bernama

commandParameternya

Parameter `command` tingkat atas menentukan apakah akan mengekspor data grafik properti atau data RDF. Jika Anda menghilangkan `command` parameter, proses ekspor default untuk mengekspor data grafik properti.

- **`export-pg`**— Ekspor data grafik properti.
- **`export-rdf`**- Ekspor data RDF.

outputS3Path Parameternya

Parameter `outputS3Path` tingkat atas diperlukan, dan harus berisi URI lokasi Amazon S3 tempat file yang diekspor dapat dipublikasikan:

```
  "outputS3Path" : "s3://(your Amazon S3 bucket)/(path to output folder)"
```

Nilai harus dimulai dengan `s3://`, diikuti dengan nama bucket yang valid dan folder path opsional dalam bucket.

`jobSize`Parameternya

Parameter `jobSize` tingkat atas hanya digunakan dengan layanan Neptune-Export, bukan dengan utilitas baris `neptune-export` perintah, dan bersifat opsional. Ini memungkinkan Anda mengkarakterisasi ukuran pekerjaan ekspor yang Anda mulai, yang membantu menentukan jumlah sumber daya komputasi yang dikhususkan untuk pekerjaan dan tingkat konkurensi maksimumnya.

```
  "jobSize" : "(one of four size descriptors)"
```

Empat deskriptor ukuran yang valid adalah:

- `small` - Konkurensi maksimum: 8. Cocok untuk volume penyimpanan hingga 10 GB.
- `medium` - Konkurensi maksimum: 32. Cocok untuk volume penyimpanan hingga 100 GB.
- `large` - Konkurensi maksimum: 64. Cocok untuk volume penyimpanan lebih dari 100 GB tapi kurang dari 1 TB.
- `xlarge` - Konkurensi maksimum: 96. Cocok untuk volume penyimpanan lebih dari 1 TB.

Secara default, ekspor yang dimulai pada layanan Neptunus-Ekspor berjalan sebagai pekerjaan. `small`

Kinerja ekspor tidak hanya bergantung pada `jobSize` pengaturan, tetapi juga pada jumlah instance database yang Anda ekspor dari, ukuran setiap instance, dan tingkat konkurensi efektif pekerjaan.

Untuk ekspor grafik properti, Anda dapat mengonfigurasi jumlah instance database menggunakan cloneClusterReplicaHitung parameter, dan Anda dapat mengonfigurasi tingkat konkurensi efektif pekerjaan menggunakan parameter. konkurensi

`params`Objeknya

Parameter `params` tingkat atas adalah objek JSON yang berisi parameter yang Anda gunakan untuk mengontrol proses ekspor itu sendiri, seperti yang dijelaskan dalam. Ekspor bidang parameter di objek `params` JSON tingkat atas Beberapa bidang dalam `params` objek khusus untuk ekspor grafik properti, beberapa ke RDF.

additionalParamsObjeknya

Parameter `additionalParams` tingkat atas adalah objek JSON yang berisi parameter yang dapat Anda gunakan untuk mengontrol tindakan yang diterapkan ke data setelah diekspor. Saat ini, hanya `additionalParams` digunakan untuk mengekspor data pelatihan untuk Neptunus ML.

Ekspor bidang parameter di objek **`params`** JSON tingkat atas

Objek JSON `params` ekspor Neptunus memungkinkan Anda untuk mengontrol ekspor, termasuk jenis dan format data yang diekspor.

Daftar bidang yang mungkin di **`params`** objek parameter ekspor

Di bawah ini adalah semua bidang tingkat atas yang mungkin muncul di `params` objek. Hanya subset dari bidang ini yang muncul di salah satu objek.

Daftar bidang yang umum untuk semua jenis ekspor

- `cloneCluster`
- `cloneClusterInstanceType`
- `cloneClusterReplicaCount`
- `cloneClusterEnableAuditLogs`
- `clusterId`
- `endpoint`
- `endpoints`
- `profile`
- `useIamAuth`
- `includeLastEventId`

Daftar bidang untuk ekspor grafik properti

- `concurrency`
- `edgeLabels`
- `filter`
- `filterConfigFile`
- `gremlinFilter`
- `gremlinNodeFilter`
- `gremlinEdgeFilter`
- `format`

- `nodeLabels`
- `scope`

Daftar bidang untuk ekspor RDF

- `format`
- `rdfExportScope`
- `sparql`
- `namedGraph`

Bidang umum untuk semua jenis ekspor

`cloneCluster`bidang di **`params`**

(Opsional). Default: `false`.

Jika `cloneCluster` parameter disetel ke`true`, proses ekspor menggunakan klon cepat dari cluster DB Anda:

```
  "cloneCluster" : true
```

Secara default, proses ekspor mengekspor data dari cluster DB yang Anda tentukan menggunakan parameter `endpoint`, `endpoints` atau `clusterId`. Namun, jika klaster DB Anda digunakan saat ekspor sedang berlangsung, dan data berubah, proses ekspor tidak dapat menjamin konsistensi data yang diekspor.

Untuk memastikan bahwa data yang diekspor konsisten, gunakan parameter `cloneCluster` untuk mengekspor dari klon statis klaster DB Anda sebagai gantinya.

Klaster DB yang diklon dibuat dalam VPC yang sama sebagai klaster DB sumber dan mewarisi grup keamanan, grup subnet dan pengaturan otentikasi database IAM sumber. Ketika ekspor selesai, Neptunus menghapus klaster DB kloning.

Secara default, klaster DB yang dikloning terdiri dari instans tunggal dari tipe instans yang sama sebagai instans utama dalam klaster DB sumber. Anda dapat mengubah tipe instans yang digunakan untuk klaster DB yang dikloning dengan menentukan satu yang berbeda menggunakan `cloneClusterInstanceType`.

> **Note**
>
> Jika Anda tidak menggunakan `cloneCluster` opsi, dan mengekspor langsung dari cluster DB utama Anda, Anda mungkin perlu meningkatkan batas waktu pada instance dari mana data sedang diekspor. Untuk set data yang besar, timeout harus diatur ke beberapa jam.

`cloneClusterInstanceType`bidang di **`params`**

(Opsional).

Jika `cloneCluster` parameter hadir dan disetel ke`true`, Anda dapat menggunakan `cloneClusterInstanceType` parameter untuk menentukan jenis instance yang digunakan untuk klaster DB kloning:

Secara default, klaster DB yang dikloning terdiri dari instans tunggal dari tipe instans yang sama sebagai instans utama dalam klaster DB sumber.

```
  "cloneClusterInstanceType" : "(for example, r5.12xlarge)"
```

`cloneClusterReplicaCount`bidang di **`params`**

(Opsional).

Jika `cloneCluster` parameter hadir dan disetel ke`true`, Anda dapat menggunakan `cloneClusterReplicaCount` parameter untuk menentukan jumlah instance baca-replika yang dibuat di klaster DB kloning:

```
  "cloneClusterReplicaCount" : (for example, 3)
```

Secara default, klaster DB yang dikloning terdiri dari instans utama tunggal. Parameter `cloneClusterReplicaCount` memungkinkan Anda menentukan berapa banyak instans baca-replika tambahan harus dibuat.

`cloneClusterEnableAuditLogs`bidang di **`params`**

(Opsional). Default: false.

Jika `cloneCluster` parameter hadir dan disetel ke true, Anda dapat menggunakan `cloneClusterEnableAuditLogs` parameter untuk mengaktifkan atau menonaktifkan log audit di kloning kloning.

Secara default, pencatatan audit dinonaktifkan.

```
"cloneClusterEnableAuditLogs" : true
```

`clusterId`bidang di **`params`**

(Opsional).

`clusterId`Parameter menentukan ID dari cluster DB untuk menggunakan:

```
  "clusterId" : "(the ID of your DB cluster)"
```

Jika Anda menggunakan `clusterId` parameter, proses ekspor menggunakan semua instance yang tersedia di cluster DB itu untuk mengekstrak data.

> **Note**
>
> Parameter `endpoint`, `endpoints`, dan `clusterId` ini sama-sama eksklusif. Gunakan satu dan satu-satunya dari mereka.

`endpoint`bidang di **`params`**

(Opsional).

Gunakan `endpoint` untuk menentukan titik akhir instans Neptunus di cluster DB Anda yang proses ekspor dapat kueri untuk mengekstrak data (lihat). Koneksi Titik akhir Ini adalah nama DNS saja, dan tidak termasuk protokol atau port:

```
  "endpoint" : "(a DNS endpoint of your DB cluster)"
```

Gunakan titik akhir cluster atau instance, tetapi bukan titik akhir pembaca utama.

> **Note**
>
> Parameter `endpoint`, `endpoints`, dan `clusterId` ini sama-sama eksklusif. Gunakan satu dan satu-satunya dari mereka.

`endpoints`bidang di **`params`**

(Opsional).

Gunakan `endpoints` untuk menentukan array titik akhir JSON di cluster DB Anda yang proses ekspor dapat kueri untuk mengekstrak data (lihatKoneksi Titik akhir). Ini hanya nama DNS, dan tidak termasuk protokol atau port:

```
"endpoints": [
  "(one endpoint in your DB cluster)",
  "(another endpoint in your DB cluster)",
  "(a third endpoint in your DB cluster)"
  ]
```

Jika Anda memiliki beberapa instans dalam klaster Anda (primer dan satu replika pembacaan atau lebih), Anda dapat meningkatkan performa ekspor dengan menggunakan parameter `endpoints` untuk mendistribusikan kueri di seluruh daftar titik akhir tersebut.

> **Note**
>
> Parameter `endpoint`, `endpoints`, dan `clusterId` ini sama-sama eksklusif. Gunakan satu dan satu-satunya dari mereka.

`profile`bidang di **`params`**

(Diperlukan untuk mengekspor data pelatihan untuk Neptunus ML, kecuali jika *`neptune_ml`* bidangnya ada di *`additionalParams`* lapangan).

`profile`Parameter ini menyediakan set parameter pra-konfigurasi untuk beban kerja tertentu. Saat ini, proses ekspor hanya mendukung `neptune_ml` profil

Jika Anda mengekspor data pelatihan untuk Neptunus ML, tambahkan parameter berikut ke objek: `params`

```
"profile" : "neptune_ml"
```

`useIamAuth`bidang di **`params`**

(Opsional). Default: `false`.

Jika database dari mana Anda mengekspor data memiliki autentikasi IAM diaktifkan, Anda harus menyertakan `useIamAuth` parameter yang disetel ke: `true`

```
"useIamAuth" : true
```

`includeLastEventId`bidang di **`params`**

Jika Anda menyetel `includeLastEventId` ke true, dan database tempat Anda mengekspor data mengaktifkan Neptunus Streams, proses ekspor akan menulis file ke lokasi ekspor `lastEventId.json` yang Anda tentukan. File ini berisi `commitNum` dan opNum dari peristiwa terakhir dalam aliran.

```
"includeLastEventId" : true
```

Database kloning yang dibuat oleh proses ekspor mewarisi pengaturan aliran induknya. Jika induk mengaktifkan aliran, klon juga akan mengaktifkan aliran. Isi aliran pada klon akan mencerminkan isi induk (termasuk peristiwa yang sama IDs) pada saat klon dibuat.

Bidang untuk ekspor grafik properti

`concurrency`bidang di **`params`**

(Opsional). Default: `4`.

`concurrency`Parameter menentukan jumlah query paralel yang harus digunakan oleh proses ekspor:

```
"concurrency" : (for example, 24)
```

Pedoman yang baik adalah mengatur tingkat konkurensi menjadi dua kali jumlah v CPUs pada semua contoh dari mana Anda mengekspor data. Contoh r5.xlarge, misalnya, memiliki 4 v. CPUs Jika Anda mengekspor dari cluster 3 instans r5.xlarge, Anda dapat mengatur tingkat konkurensi ke 24 (= 3 x 2 x 4).

Jika Anda menggunakan layanan Neptune-Export, tingkat konkurensi dibatasi oleh pengaturan jobSize. Sebuah pekerjaan kecil, misalnya, mendukung tingkat konkurensi 8. Jika Anda mencoba menentukan tingkat konkurensi 24 untuk pekerjaan kecil menggunakan `concurrency` parameter, level efektif tetap pada 8.

Jika Anda mengekspor dari klaster yang dikloning, proses ekspor menghitung tingkat konkurensi yang tepat berdasarkan ukuran instans yang dikloning dan ukuran pekerjaan.

`edgeLabels`bidang di **`params`**

(Opsional).

Gunakan `edgeLabels` untuk mengekspor hanya tepi yang memiliki label yang Anda tentukan:

```
  "edgeLabels" : ["(a label)", "(another label"]
```

Setiap label dalam JSON array harus satu, label sederhana.

`scope`Parameter lebih diutamakan daripada `edgeLabels` parameter, jadi jika `scope` nilainya tidak termasuk tepi, `edgeLabels` parameter tidak berpengaruh.

`filter`bidang di **`params`**

(Opsional).

Gunakan `filter` untuk menentukan bahwa hanya and/or tepi node dengan label tertentu yang harus diekspor, dan untuk memfilter properti yang diekspor untuk setiap node atau tepi.

Struktur umum suatu `filter` objek, baik inline atau dalam file filter-konfigurasi, adalah sebagai berikut:

```
  "filter" : {
    "nodes": [ (array of node label and properties objects) ],
    "edges": [ (array of edge definition an properties objects) ]
  }
```

- **`nodes`**— Berisi array JSON simpil dan properti simpul dalam bentuk berikut:

  ```
      "nodes : [
        {
          "label": "(node label)",
          "properties": [ "(a property name)", "(another property name)", ( ... ) ]
        }
      ]
  ```

  - `label` — Label atau label properti-grafik node.

Mengambil nilai tunggal atau, jika node memiliki beberapa label, array nilai.

- `properties`— Berisi array nama-nama properti node yang ingin Anda ekspor.

- **`edges`** — Berisi array JSON definisi edge dalam bentuk berikut:

```
"edges" : [
  {
    "label": "(edge label)",
    "properties": [ "(a property name)", "(another property name)", ( ... ) ]
  }
]
```

  - `label`— Label grafik properti edge. Mengambil satu nilai.
  - `properties`— Berisi array nama-nama properti edge yang ingin Anda ekspor.

`filterConfigFile`bidang di **`params`**

(Opsional).

Gunakan `filterConfigFile` untuk menentukan file JSON yang berisi konfigurasi filter dalam bentuk yang sama dengan `filter` parameter:

```
"filterConfigFile" : "s3://(your Amazon S3 bucket)/neptune-export/(the name of the JSON file)"
```

Lihat filter untuk format `filterConfigFile` file.

`format`bidang yang digunakan untuk data grafik properti di **`params`**

(Opsional). Default: `csv` (nilai yang dipisahkan koma)

`format`Parameter menentukan format output dari data grafik properti yang diekspor:

```
"format" : (one of: csv, csvNoHeaders, json, neptuneStreamsJson)
```

- **`csv`**— Nilai dipisahkan koma (CSV) output diformat, dengan judul kolom diformat sesuai dengan format data beban Gremlin.
- **`csvNoHeaders`**— Data berformat CSV tanpa judul kolom.

- **`json`**— Data yang diformat JSON.
- **`neptuneStreamsJson`**— Data berformat JSON yang menggunakan format serialisasi perubahan GREMLIN_JSON.

`gremlinFilter`bidang di **`params`**

(Opsional).

`gremlinFilter`Parameter ini memungkinkan Anda untuk menyediakan cuplikan Gremlin, seperti `has()` langkah, yang digunakan untuk memfilter kedua node dan tepi:

```
  "gremlinFilter" : (a Gremlin snippet)
```

Nama bidang dan nilai string harus dikelilingi oleh tanda kutip ganda yang lolos. Untuk tanggal dan waktu, Anda dapat menggunakan metode datetime.

Contoh berikut hanya mengekspor node dan tepi dengan properti yang dibuat tanggal yang nilainya lebih besar dari 2021-10-10:

```
  "gremlinFilter" : "has(\"created\", gt(datetime(\"2021-10-10\")))"
```

`gremlinNodeFilter`bidang di **`params`**

(Opsional).

`gremlinNodeFilter`Parameter ini memungkinkan Anda untuk menyediakan cuplikan Gremlin, seperti `has()` langkah, yang digunakan untuk memfilter node:

```
  "gremlinNodeFilter" : (a Gremlin snippet)
```

Nama bidang dan nilai string harus dikelilingi oleh tanda kutip ganda yang lolos. Untuk tanggal dan waktu, Anda dapat menggunakan metode datetime.

Contoh berikut mengekspor hanya node dengan properti `deleted` Boolean yang nilainya adalah: `true`

```
  "gremlinNodeFilter" : "has(\"deleted\", true)"
```

`gremlinEdgeFilter`bidang di **`params`**

(Opsional).

`gremlinEdgeFilter`Parameter ini memungkinkan Anda untuk menyediakan cuplikan Gremlin, seperti `has()` langkah, yang digunakan untuk menyaring tepi:

```
  "gremlinEdgeFilter" : (a Gremlin snippet)
```

Nama bidang dan nilai string harus dikelilingi oleh tanda kutip ganda yang lolos. Untuk tanggal dan waktu, Anda dapat menggunakan metode datetime.

Contoh berikut hanya mengekspor tepi dengan properti `strength` numerik yang nilainya 5:

```
  "gremlinEdgeFilter" : "has(\"strength\", 5)"
```

`nodeLabels`bidang di **`params`**

(Opsional).

Gunakan `nodeLabels` untuk mengekspor hanya node yang memiliki label yang Anda tentukan:

```
  "nodeLabels" : ["(a label)", "(another label"]
```

Setiap label dalam JSON array harus satu, label sederhana.

`scope`Parameter lebih diutamakan daripada `nodeLabels` parameter, jadi jika `scope` nilainya tidak termasuk node, `nodeLabels` parameter tidak berpengaruh.

`scope`bidang di **`params`**

(Opsional). Default: `all`.

`scope`Parameter menentukan apakah untuk mengekspor hanya node, atau hanya tepi, atau kedua node dan tepi:

```
  "scope" : (one of: nodes, edges, or all)
```

- `nodes` — Ekspor simpul dan propertinya saja.

- `edges` — Ekspor edge dan propertinya saja.
- `all` — Ekspor simpul dan edge dan propertinya (default).

Bidang untuk ekspor RDF

`format`bidang yang digunakan untuk data RDF di **`params`**

(Opsional). Default: `turtle`

`format`Parameter menentukan format output dari data RDF yang diekspor:

```
  "format" : (one of: turtle, nquads, ntriples, neptuneStreamsJson)
```

- **`turtle`**— Output yang diformat kura-kura.
- **`nquads`**— N-Quads diformat data tanpa judul kolom.
- **`ntriples`**— N-Triples data yang diformat.
- **`neptuneStreamsJson`**— Data berformat JSON yang menggunakan format serialisasi perubahan SPARQL NQUADS.

`rdfExportScope`bidang di **`params`**

(Opsional). Default: `graph`.

`rdfExportScope`Parameter menentukan ruang lingkup ekspor RDF:

```
  "rdfExportScope" : (one of: graph, edges, or query)
```

- `graph`— Ekspor semua data RDF.
- `edges`— Ekspor hanya tiga kali lipat yang mewakili tepi.
- `query`— Ekspor data diambil oleh kueri SPARQL yang disediakan menggunakan bidang. `sparql`

`sparql`bidang di **`params`**

(Opsional).

`sparql`Parameter ini memungkinkan Anda menentukan kueri SPARQL untuk mengambil data yang akan diekspor:

```
  "sparql" : (a SPARQL query)
```

Jika Anda menyediakan kueri menggunakan `sparql` bidang, Anda juga harus mengatur `rdfExportScope` bidang kequery.

`namedGraph`bidang di **`params`**

(Opsional).

namedGraphParameter ini memungkinkan Anda untuk menentukan IRI untuk membatasi ekspor ke grafik bernama tunggal:

```
  "namedGraph" : (Named graph IRI)
```

namedGraphParameter hanya dapat digunakan dengan `rdfExportScope` bidang yang disetel kegraph.

Contoh penyaringan apa yang diekspor

Berikut adalah contoh yang menggambarkan cara untuk memfilter data yang diekspor.

Memfilter ekspor data grafik properti

Contoh penggunaan **`scope`** untuk mengekspor hanya tepi

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "scope": "edges"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Contoh menggunakan **`nodeLabels`** dan **`edgeLabels`** mengekspor hanya node dan tepi yang memiliki label tertentu

Parameter `nodeLabels` dalam contoh berikut menentukan bahwa hanya simpul yang memiliki label `Person` atau label `Post` harus diekspor. Parameter `edgeLabels` menentukan bahwa hanya edge dengan label `likes` harus diekspor:

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "nodeLabels": ["Person", "Post"],
    "edgeLabels": ["likes"]
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Contoh penggunaan **`filter`** untuk mengekspor hanya node, tepi, dan properti tertentu

`filter`Objek dalam contoh ini mengekspor `country` node dengan`type`, `code` dan `desc` propertinya, dan juga `route` tepi dengan `dist` propertinya.

```
{
  "command": "export-pg",
  "params": {
```

```
    "endpoint": "(your Neptune endpoint DNS name)",
    "filter": {
      "nodes": [
        {
          "label": "country",
          "properties": [
            "type",
            "code",
            "desc"
          ]
        }
      ],
      "edges": [
        {
          "label": "route",
          "properties": [
            "dist"
          ]
        }
      ]
    }
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Contoh yang menggunakan **`gremlinFilter`**

Contoh ini hanya digunakan `gremlinFilter` untuk mengekspor node dan tepi yang dibuat setelah 2021-10-10 (yaitu, dengan `created` properti yang nilainya lebih besar dari 2021-10-10):

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinFilter" : "has(\"created\", gt(datetime(\"2021-10-10\")))"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Contoh yang menggunakan **`gremlinNodeFilter`**

Contoh ini digunakan `gremlinNodeFilter` untuk mengekspor hanya node yang dihapus (node dengan `deleted` properti Boolean yang nilainya`true`):

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinNodeFilter" : "has(\"deleted\", true)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Contoh yang menggunakan **`gremlinEdgeFilter`**

Contoh ini digunakan `gremlinEdgeFilter` untuk mengekspor hanya tepi dengan properti `strength` numerik yang nilainya 5:

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinEdgeFilter" : "has(\"strength\", 5)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Menggabungkan**`filter`**,**`gremlinNodeFilter`**,**`nodeLabels`**, **`edgeLabels`** dan **`scope`**

`filter`Objek dalam contoh ini mengekspor:

- `country`node dengan merekatype, code dan `desc` properti
- `airport`node dengan merekacode, `icao` dan `runways` properti
- `route`tepi dengan `dist` properti mereka

`gremlinNodeFilter`Parameter menyaring node sehingga hanya node dengan `code` properti yang nilainya dimulai dengan A yang diekspor.

`edgeLabels`Parameter `nodeLabels` dan selanjutnya membatasi output sehingga hanya `airport` node dan `route` tepi yang diekspor.

Akhirnya, `scope` parameter menghilangkan tepi dari ekspor, yang hanya menyisakan `airport` node yang ditunjuk dalam output.

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "filter": {
      "nodes": [
        {
          "label": "airport",
          "properties": [
            "code",
            "icao",
            "runways"
          ]
        },
        {
          "label": "country",
          "properties": [
            "type",
            "code",
            "desc"
          ]
        }
      ],
      "edges": [
        {
          "label": "route",
          "properties": [
            "dist"
          ]
        }
      ]
    },
    "gremlinNodeFilter": "has(\"code\", startingWith(\"A\"))",
    "nodeLabels": [
      "airport"
    ],
    "edgeLabels": [
      "route"
    ],
    "scope": "nodes"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Memfilter ekspor data RDF

Menggunakan **`rdfExportScope`** dan **`sparql`** mengekspor tepi tertentu

Contoh ini mengekspor tiga kali lipat yang predikatnya < http://kelvinlawrence. net/air-routes/objectProperty/route> dan yang objeknya tidak literal:

```
{
  "command": "export-rdf",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "rdfExportScope": "query",
    "sparql": "CONSTRUCT { ?s <http://kelvinlawrence.net/air-routes/objectProperty/
route> ?o } WHERE { ?s ?p ?o . FILTER(!isLiteral(?o)) }"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Menggunakan **`namedGraph`** untuk mengekspor satu grafik bernama

Contoh ini mengekspor tiga kali lipat milik grafik bernama < http://aws.amazon. com/neptune/vocab/v01/DefaultNamedGraph>:

```
{
  "command": "export-rdf",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "rdfExportScope": "graph",
    "namedGraph": "http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Memecahkan masalah proses ekspor Neptunus

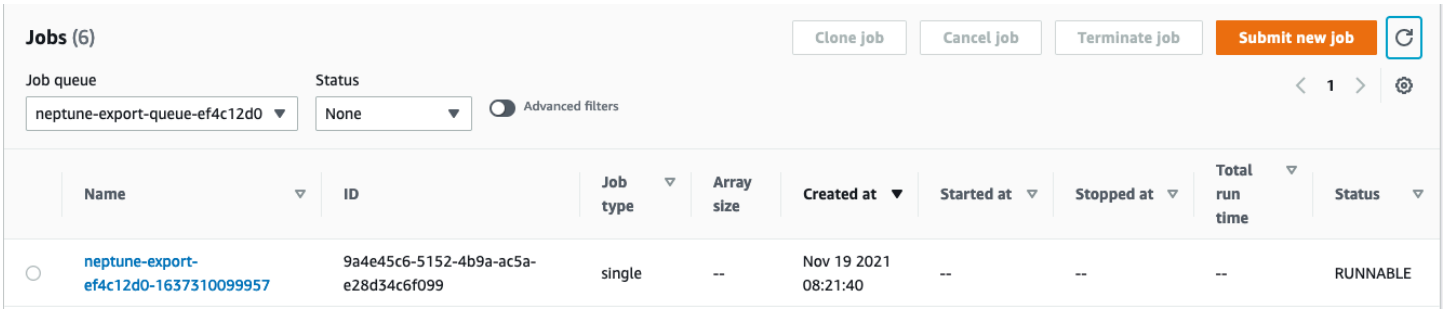
Proses ekspor Amazon Neptunus AWS Batchdigunakan untuk menyediakan sumber daya komputasi dan penyimpanan yang diperlukan untuk mengekspor data Neptunus Anda. Saat ekspor sedang berjalan, Anda dapat menggunakan tautan di `logs` bidang untuk mengakses CloudWatch log untuk pekerjaan ekspor.

Namun, CloudWatch log untuk AWS Batch pekerjaan yang melakukan ekspor hanya tersedia saat AWS Batch pekerjaan sedang berjalan. Jika ekspor Neptunus melaporkan bahwa ekspor dalam status tertunda, tidak akan ada tautan log tempat Anda dapat mengakses log. CloudWatch Jika pekerjaan ekspor tetap berada di `pending` negara bagian selama lebih dari beberapa menit, mungkin ada masalah dalam penyediaan sumber daya yang mendasarinya AWS Batch .

Ketika pekerjaan ekspor meninggalkan status tertunda, Anda dapat memeriksa statusnya sebagai berikut:

Untuk memeriksa status AWS Batch pekerjaan

1. Buka AWS Batch konsol di https://console.aws.amazon.com/batch/.
2. Pilih antrian pekerjaan neptune-export.
3. Cari pekerjaan yang namanya cocok dengan ekspor yang `jobName` dikembalikan oleh Neptunus saat Anda memulai ekspor.

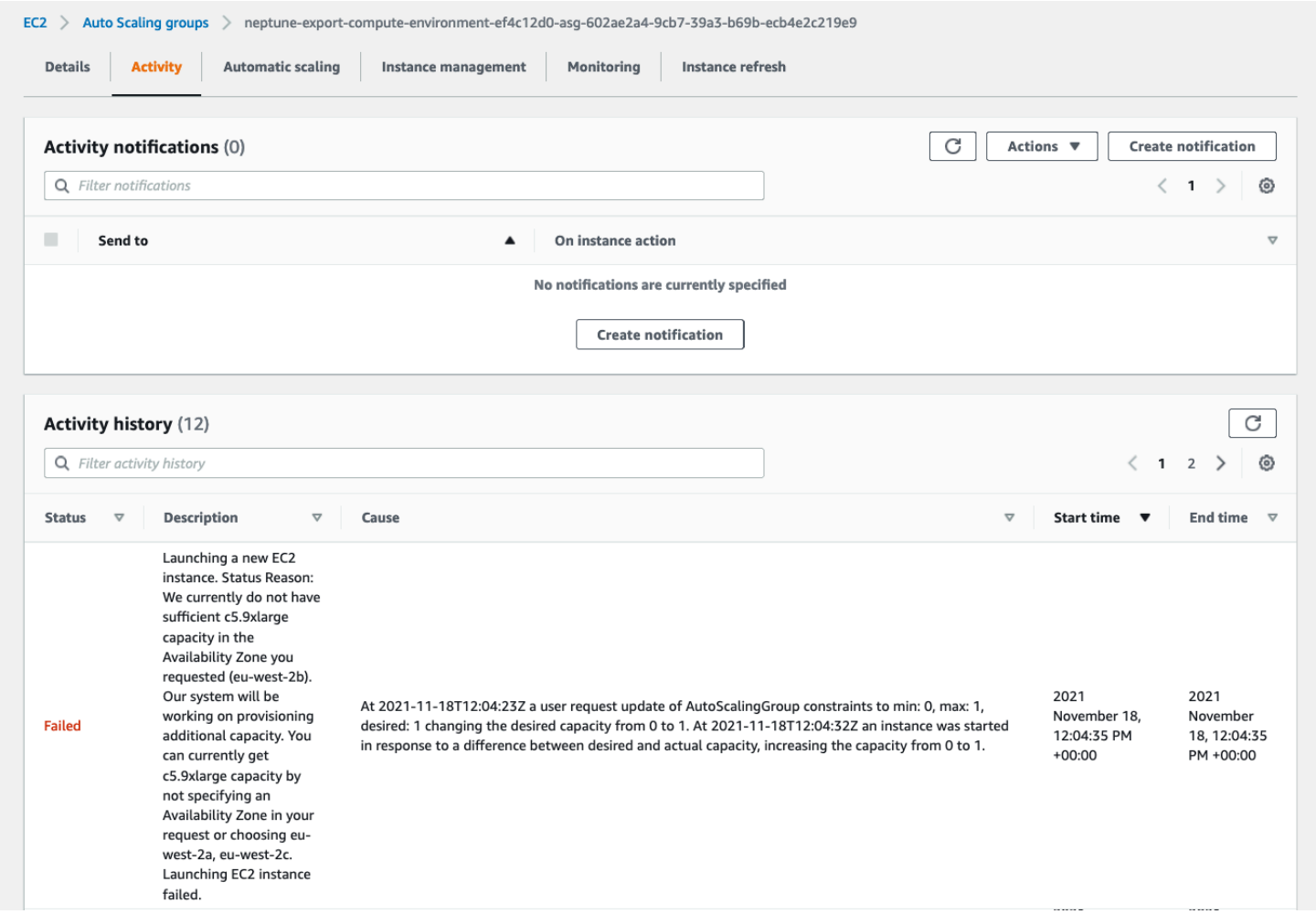


Jika pekerjaan tetap macet dalam `RUNNABLE` status, itu mungkin karena masalah jaringan atau keamanan mencegah instance container bergabung dengan cluster Amazon Elastic Container Service (Amazon ECS) yang mendasarinya. Lihat bagian tentang memverifikasi pengaturan jaringan dan keamanan lingkungan komputasi di artikel dukungan ini.

Hal lain yang dapat Anda periksa adalah masalah dengan auto-scaling:

Untuk memeriksa grup EC2 auto-scaling Amazon untuk lingkungan komputasi AWS Batch

1. Buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
2. Pilih grup Auto Scaling untuk lingkungan komputasi neptune-export.
3. Buka tab Aktivitas dan periksa riwayat aktivitas untuk peristiwa yang tidak berhasil.



Neptune Export kesalahan umum

`org.eclipse.rdf4j.query.QueryEvaluationException: Tag mismatch!`

Jika suatu `export-rdf` pekerjaan secara teratur gagal dengan a `Tag mismatch!QueryEvaluationException`, instance Neptunus berukuran terlalu kecil untuk kueri besar yang berjalan lama yang digunakan oleh Ekspor Neptunus.

Anda dapat menghindari kesalahan ini dengan meningkatkan ke instance Neptunus yang lebih besar atau dengan mengonfigurasi pekerjaan untuk mengekspor dari cluster kloning besar, seperti ini:

```
'{
  "command": "export-rdf",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
```

```
    "cloneCluster": True,
    "cloneClusterInstanceType" : "r5.24xlarge"
  }
}'
```

Mengekspor hasil kueri Gremlin ke Amazon S3

Mulai rilis mesin 1.4.3.0, Amazon Neptunus mendukung ekspor hasil kueri Gremlin langsung ke Amazon S3. Fitur ini memungkinkan Anda menangani hasil kueri besar secara efisien dengan mengekspornya ke bucket Amazon S3 alih-alih mengembalikannya sebagai respons kueri.

Untuk mengekspor hasil kueri ke Amazon S3, gunakan `call()` langkah dengan nama `neptune.query.exportToS3` layanan sebagai langkah terakhir dalam kueri Gremlin Anda. Langkah terminal di driver Tinkerpop menggunakan Bytecode dapat ditambahkan setelah langkah. `call()` Parameter ekspor harus disediakan sebagai nilai string.

> **Note**
>
> Kueri dengan `call()` langkah memiliki `neptune.query.exportToS3` akan gagal jika tidak digunakan sebagai langkah terakhir. Klien Gremlin menggunakan bytecode dapat menggunakan langkah-langkah terminal. Lihat praktik terbaik Gremlin dalam dokumentasi Amazon Neptunus untuk informasi selengkapnya.

```
g.V()
  ...
  .call('neptune.query.exportToS3', [
    'destination': 's3://your-bucket/path/result.json',
    'format': 'GraphSONv3',
    'kmskeyArn': 'optional-kms-key-arn'
  ])
```

Parameter

- `destination`: wajib - URI Amazon S3 tempat hasil akan ditulis.
- `format`: required - Format output, saat ini hanya mendukung 'Grafik SONv3'.
- `keyArn`: opsional - ARN AWS KMS kunci untuk enkripsi sisi server Amazon S3.

Contoh

Contoh kueri

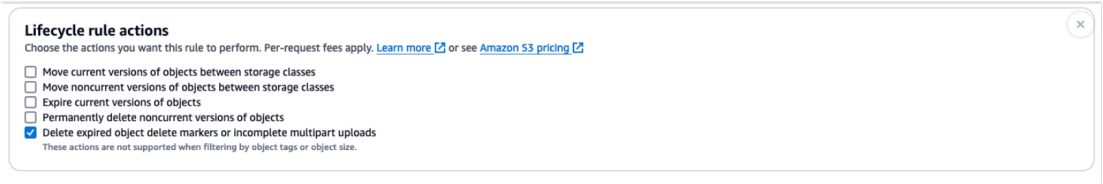
```
g.V().
    hasLabel('Comment').
    valueMap().
    call('neptune.query.exportToS3', [
    'destination': 's3://your-bucket/path/result.json',
    'format': 'GraphSONv3',
    'keyArn': 'optional-kms-key-arn'
  ])
```

Contoh respons kueri

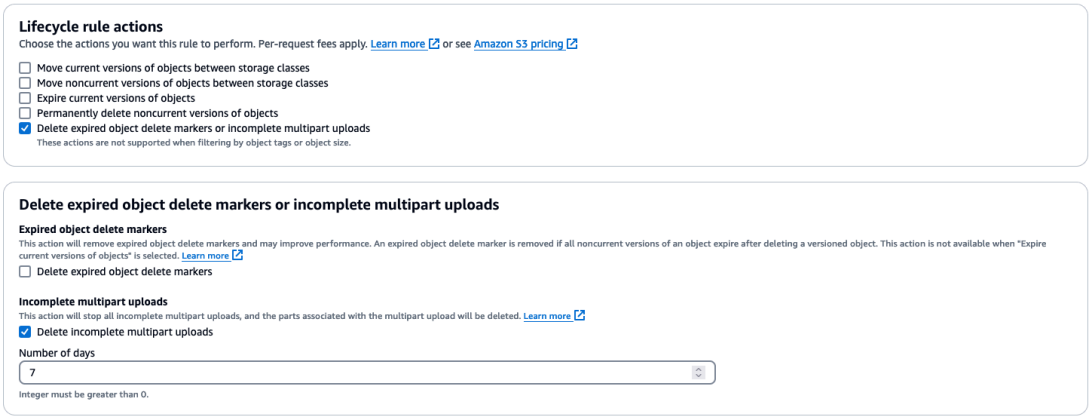
```
{
    "destination":"s3://your-bucket/path/result.json,
    "exportedResults": 100,
    "exportedBytes": 102400
}
```

Prasyarat

- Instans DB Neptunus Anda harus memiliki akses ke Amazon S3 melalui titik akhir VPC dari jenis gateway.
- Untuk menggunakan AWS KMS enkripsi kustom dalam kueri, titik akhir VPC tipe antarmuka untuk diperlukan agar AWS KMS Neptunus dapat berkomunikasi dengannya. AWS KMS
- Anda harus mengaktifkan autentikasi IAM di Neptunus, dan memiliki izin IAM yang sesuai untuk menulis ke bucket Amazon S3 target. Tidak memiliki ini akan menyebabkan 400 kesalahan permintaan buruk “Cluster harus mengaktifkan otentikasi IAM untuk Ekspor S3”.
- Target ember Amazon S3:
  - Bucket Amazon S3 target tidak boleh bersifat publik. `Block public access`harus diaktifkan.
  - Target tujuan Amazon S3 harus kosong.
  - Bucket Amazon S3 target harus memiliki aturan siklus hidup. `Delete expired object delete markers or incomplete multipart uploads` `Delete incomplete multipart uploads` Lihat pembaruan manajemen siklus hidup Amazon S3 - dukungan untuk pengunggahan multibagian dan hapus penanda untuk informasi selengkapnya.



- Bucket Amazon S3 target harus mengaktifkan aturan siklus hidup `Delete expired object delete markers or incomplete multipart uploads` dengan `Delete incomplete multipart uploads` disetel ke nilai yang lebih tinggi dari evaluasi kueri (mis., 7 hari). Ini diperlukan untuk menghapus unggahan yang tidak lengkap (yang tidak terlihat secara langsung tetapi akan dikenakan biaya) jika tidak dapat diselesaikan atau dibatalkan oleh Neptunus (misalnya, karena kegagalan). instance/engine Lihat pembaruan manajemen siklus hidup Amazon S3 - dukungan untuk pengunggahan multibagian dan hapus penanda untuk informasi selengkapnya.



Pertimbangan penting

- Langkah ekspor harus menjadi langkah terakhir dalam kueri Gremlin Anda.
- Jika objek sudah ada di lokasi Amazon S3 yang ditentukan, kueri akan gagal.
- Waktu eksekusi kueri maksimum untuk kueri ekspor dibatasi hingga 11 jam 50 menit. Fitur ini menggunakan sesi akses Teruskan. Saat ini dibatasi hingga 11 jam dan 50 menit untuk menghindari masalah kedaluwarsa token.

> **Note**
>
> Kueri ekspor masih menghormati batas waktu kueri. Untuk ekspor besar, Anda harus menggunakan batas waktu kueri yang sesuai.

- Semua unggahan objek baru ke Amazon S3 secara otomatis dienkripsi.

- Untuk menghindari biaya penyimpanan dari unggahan multibagian yang tidak lengkap jika terjadi kesalahan atau crash, sebaiknya atur aturan siklus hidup dengan bucket Amazon S3 Anda`Delete incomplete multipart uploads`.

Format respons

Daripada mengembalikan hasil kueri secara langsung, kueri mengembalikan metadata tentang operasi ekspor, termasuk detail status dan ekspor. Hasil kueri di Amazon S3 akan dalam format Grafik SONv3.


```
{
  "data": {
    "@type": "g:List",
    "@value": [
      {
        "@type": "g:Map",
        "@value": [
          "browserUsed",
          {
            "@type": "g:List",
            "@value": [
              "Safari"
            ]
          },
          "length",
          {
            "@type": "g:List",
            "@value": [
              {
                "@type": "g:Int32",
                "@value": 7
              }
            ]
          },
          "locationIP",
          {
            "@type": "g:List",
            "@value": [
              "192.0.2.0/24"
            ]
          },
          "creationDate",
```

```
          {
            "@type": "g:List",
            "@value": [
              {
                "@type": "g:Date",
                "@value": 1348341961000
              }
            ]
          },
          "content",
          {
            "@type": "g:List",
            "@value": [
              "no way!"
            ]
          }
        ]
      },
      {
        "@type": "g:Map",
        "@value": [
          "browserUsed",
          {
            "@type": "g:List",
            "@value": [
              "Firefox"
            ]
          },
          "length",
          {
            "@type": "g:List",
            "@value": [
              {
                "@type": "g:Int32",
                "@value": 2
              }
            ]
          },
          "locationIP",
          {
            "@type": "g:List",
            "@value": [
              "203.0.113.0/24"
            ]
```

```
          },
          "creationDate",
          {
            "@type": "g:List",
            "@value": [
              {
                "@type": "g:Date",
                "@value": 1348352960000
              }
            ]
          },
          "content",
          {
            "@type": "g:List",
            "@value": [
              "ok"
            ]
          }
        ]
      },


      ...


    ]
  }
}
```

Keamanan

- Semua data yang ditransfer ke Amazon S3 dienkripsi dalam perjalanan menggunakan SSL.
- Anda dapat menentukan AWS KMS kunci untuk enkripsi sisi server dari data yang diekspor. Amazon S3 mengenkripsi data baru secara default. Jika bucket dikonfigurasi untuk menggunakan AWS KMS kunci tertentu, maka kunci itu digunakan.
- Neptunus memverifikasi bahwa ember target tidak bersifat publik sebelum memulai ekspor.
- Ekspor lintas akun dan lintas wilayah tidak didukung.

Penanganan kesalahan

- Target ember Amazon S3 adalah publik.

- Objek yang ditentukan sudah ada.
- Anda tidak memiliki izin yang cukup untuk menulis ke bucket Amazon S3.
- Eksekusi kueri melebihi batas waktu maksimum.

Praktik terbaik

- Gunakan aturan siklus hidup bucket Amazon S3 untuk membersihkan unggahan multibagian yang tidak lengkap.
- Pantau operasi ekspor Anda menggunakan log dan metrik Neptunus. Anda dapat memeriksa titik akhir status Gremlin untuk melihat apakah kueri sedang berjalan. Selama klien belum menerima respons, kueri akan diasumsikan berjalan.

Memberikan akses untuk fitur ekspor Amazon S3 Gremlin

Kebijakan IAM yang diperlukan

1. Akses baca kueri Neptunus

```
{
  "Sid": "NeptuneQueryRead",
  "Effect": "Allow",
  "Action": ["neptune-db:Read*"],
  "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-ABCD12/*"
}
```

   Mengapa diperlukan: Izin ini memungkinkan membaca data dari database Neptunus, yang diperlukan untuk mengeksekusi kueri Gremlin yang akan diekspor. Contoh sebelumnya memungkinkan kueri baca. Untuk read/write kueri, write/delete izin diperlukan.

2. Izin ekspor Amazon S3

```
{
  "Sid": "NeptuneS3Export",
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:PutObject",
    "s3:AbortMultipartUpload",
    "s3:GetBucketPublicAccessBlock"
```

```
  ],
  "Resource": "arn:aws:s3:::neptune-export-bucket/*"
}
```

Mengapa setiap izin diperlukan:

- `s3:ListBucket`: Diperlukan untuk memverifikasi keberadaan bucket dan daftar konten.
- `s3:PutObject`: Diperlukan untuk menulis data yang diekspor ke Amazon S3.
- `s3:AbortMultipartUpload`: Diperlukan untuk membersihkan unggahan multipart yang tidak lengkap jika ekspor gagal.
- `s3:GetBucketPublicAccessBlock`: Diperlukan sebagai langkah pengamanan untuk memverifikasi bahwa bucket tidak bersifat publik sebelum mengekspor data.

3. AWS KMSpermissios - opsional. Hanya diperlukan jika menggunakan AWS KMS enkripsi khusus.

```
{
  "Sid": "NeptuneS3ExportKMS",
  "Effect": "Allow",
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey",
    "kms:DescribeKey"
  ],
  "Resource": "arn:aws:kms:<REGION>:<AWS_ACCOUNT_ID>:key/mrk-48971c37"
    "Condition": {
    "StringEquals": {
      "kms:ViaService": [
        "s3.<REGION>.amazonaws.com",
        "rds.<REGION>.amazonaws.com"
      ]
    }
  }
}
```

Mengapa setiap izin diperlukan:

- `kms:Decrypt`: Diperlukan untuk mendekripsi AWS KMS kunci untuk enkripsi data.
- `kms:GenerateDataKey`: Diperlukan untuk menghasilkan kunci data untuk mengenkripsi data yang diekspor.
- `kms:DescribeKey`: Diperlukan untuk memverifikasi dan mengambil informasi tentang AWS KMS kunci.

- `kms:ViaService`: Meningkatkan keamanan dengan menegakkan bahwa kunci tidak dapat digunakan oleh peran ini untuk AWS layanan lain.

Prasyarat penting

- Autentikasi IAM: Harus diaktifkan pada cluster Neptunus untuk menegakkan izin ini.
- Titik akhir VPC:
  - Titik akhir VPC tipe Gateway untuk Amazon S3 diperlukan untuk memungkinkan Neptunus berkomunikasi dengan Amazon S3.
  - Untuk menggunakan AWS KMS enkripsi kustom dalam kueri, titik akhir VPC tipe antarmuka untuk diperlukan agar AWS KMS Neptunus dapat berkomunikasi dengannya. AWS KMS
- Konfigurasi bucket Amazon S3:
  - Tidak harus publik.
  - Harus memiliki aturan siklus hidup untuk membersihkan unggahan multibagian yang tidak lengkap.
  - Akan secara otomatis mengenkripsi objek baru.

Izin dan prasyarat ini memastikan ekspor hasil kueri Gremlin yang aman dan andal sambil mempertahankan kontrol akses dan tindakan perlindungan data yang tepat.

Mengelola Basis Data Amazon Neptune Anda

Bagian ini menunjukkan cara mengelola dan memelihara cluster DB Neptunus Anda menggunakan Konsol Manajemen AWS dan. AWS CLI

Neptune beroperasi di klaster server basis data yang terhubung dalam topologi replikasi. Dengan demikian, mengelola Neptunus sering melibatkan penerapan perubahan ke beberapa server dan memastikan bahwa semua replika Neptunus mengikuti server utama.

Karena Neptune secara transparan menskalakan penyimpanan yang mendasari saat data Anda tumbuh, pengelolaan Neptune memerlukan manajemen penyimpanan disk yang relatif kecil. Demikian pula, karena Neptune secara otomatis melakukan pencadangan berkelanjutan, klaster Neptune tidak memerlukan perencanaan atau waktu henti yang ekstensif untuk melakukan pencadangan.

Topik

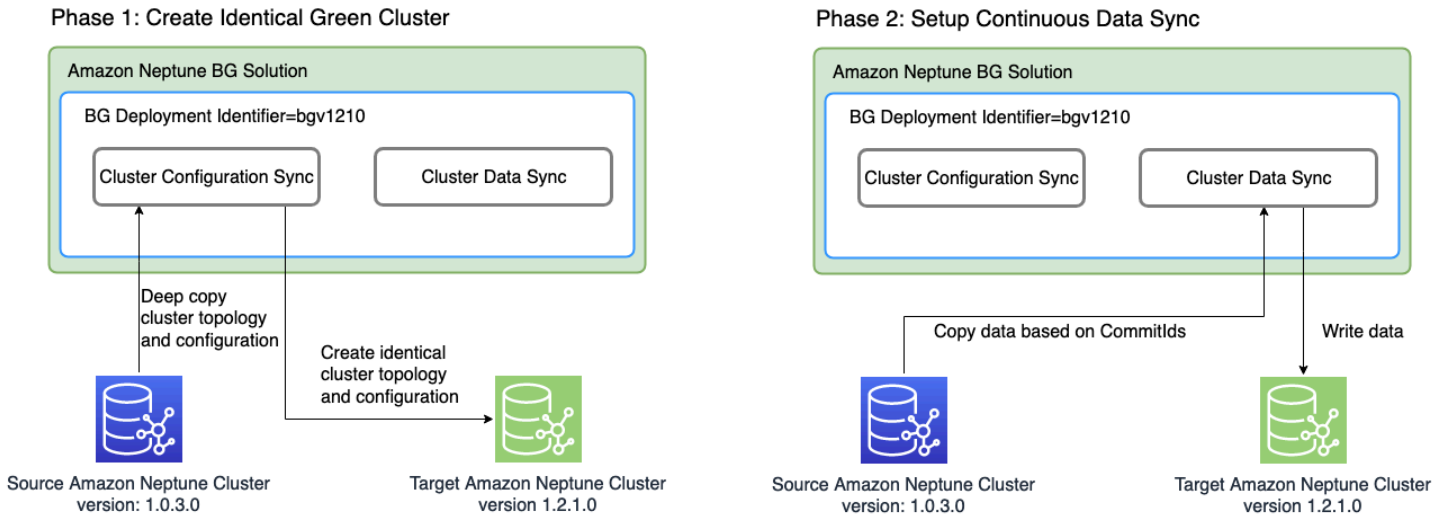
- Menggunakan solusi Blue/Green Neptunus untuk melakukan pembaruan biru-hijau
- Membuat pengguna IAM dengan izin untuk Neptunus
- Grup parameter Amazon Neptunus
- Parameter Amazon Neptunus
- Meluncurkan cluster DB Neptunus menggunakan Konsol Manajemen AWS
- Menghentikan dan memulai cluster DB Amazon Neptunus
- Kosongkan klaster DB Amazon Neptune menggunakan API reset cepat
- Menambahkan instance pembaca Neptunus ke Cluster DB
- Membuat instance pembaca Neptunus menggunakan konsol
- Memodifikasi Klaster DB Neptune Menggunakan Konsol
- Performa dan Penskalaan di Amazon Neptune
- Penskalaan otomatis jumlah replika di cluster DB Amazon Neptunus
- Mempertahankan Cluster DB Amazon Neptunus Anda
- Menggunakan CloudFormation template untuk memperbarui versi mesin Cluster DB Neptunus Anda
- Kloning Basis Data di Neptune
- Mengelola Instans Amazon Neptune

Menggunakan solusi Blue/Green Neptunus untuk melakukan pembaruan biru-hijau

Peningkatan mesin Amazon Neptunus dapat memerlukan waktu henti aplikasi karena database tidak tersedia saat pembaruan sedang diinstal dan diverifikasi. Ini benar apakah mereka dimulai secara manual atau otomatis.

Neptunus menyediakan solusi penerapan Blue/Green yang dapat Anda jalankan menggunakan tumpukan dan CloudFormation sangat mengurangi waktu henti tersebut. Ini menciptakan lingkungan pementasan hijau yang disinkronkan dengan lingkungan produksi biru Anda. Anda kemudian dapat memperbarui lingkungan pementasan tersebut untuk melakukan peningkatan versi mesin kecil atau utama, perubahan model data grafik, atau pembaruan sistem operasi, dan menguji hasilnya. Akhirnya, Anda dapat mengubahnya dengan cepat untuk menjadi lingkungan produksi Anda, dengan waktu henti yang sangat sedikit.

Solusi Blue/Green Neptunus melewati dua fase, seperti yang diilustrasikan dalam diagram ini:



Fase 1 membuat cluster DB Hijau yang identik dengan cluster produksi Anda

Solusinya membuat cluster DB dengan pengenal blue/green penerapan unik dan dengan topologi cluster yang sama dengan cluster produksi Anda. Artinya, ia memiliki jumlah dan ukuran instans DB yang sama, grup parameter yang sama dan semua konfigurasi yang sama dengan cluster DB produksi (biru) kecuali bahwa itu telah ditingkatkan ke versi mesin target yang Anda tentukan, yang harus lebih tinggi dari versi mesin (biru) Anda saat ini. Anda dapat menentukan versi mesin minor dan utama untuk target. Jika perlu, solusi akan melakukan upgrade menengah yang diperlukan untuk mencapai versi mesin target yang ditentukan. Cluster baru ini menjadi lingkungan pementasan hijau.

Tahap 2 mengatur sinkronisasi data berkelanjutan

Setelah lingkungan hijau sepenuhnya disiapkan, solusinya mengatur replikasi berkelanjutan antara cluster sumber (biru) dan cluster target (hijau) menggunakan aliran Neptunus. Ketika perbedaan replikasi di antara mereka mencapai nol, lingkungan pementasan siap untuk pengujian. Pada saat itu Anda harus menjeda penulisan ke cluster biru untuk menghindari kelambatan replikasi lebih lanjut.

Versi mesin target Anda mungkin memiliki fitur atau dependensi baru yang memengaruhi aplikasi Anda. Periksa halaman rilis mesin target dan halaman rilis mesin intervensi di bawah Rilis mesin untuk melihat apa yang telah berubah sejak versi mesin Anda saat ini. Yang terbaik adalah menjalankan pengujian integrasi atau memverifikasi aplikasi Anda secara manual di klaster hijau sebelum mempromosikannya ke lingkungan produksi.

Setelah Anda menguji dan memenuhi syarat perubahan di cluster hijau, cukup alihkan titik akhir database dalam aplikasi Anda dari biru ke cluster hijau.

Setelah peralihan, Blue/Green solusi Neptunus tidak menghapus lingkungan produksi biru lama. Anda masih akan memiliki akses ke sana untuk validasi dan pengujian tambahan jika diperlukan. Biaya penagihan standar berlaku untuk instance-nya sampai Anda menghapusnya. Blue/Green Solusinya juga menggunakan AWS layanan lain, biaya yang ditagih dengan harga normal. Detail tentang menghapus solusi ketika Anda selesai dengan itu tercakup di bagian pembersihan.

Prasyarat untuk menjalankan tumpukan Neptunus Blue/Green

Sebelum meluncurkan tumpukan Blue/Green Neptunus:

- Pastikan untuk mengaktifkan aliran Neptunus di cluster produksi (biru) Anda.
- Semua instance di cluster biru Anda harus dalam keadaan tersedia. Anda dapat memeriksa status instance di konsol Neptunus atau dengan menggunakan API. describe-db-instances
- Semua instance juga harus sinkron dengan grup parameter cluster DB.
- Solusi Blue/Green Neptunus memerlukan titik akhir DynamoDB VPC di VPC tempat cluster biru Anda berada. Lihat Menggunakan titik akhir Amazon VPC untuk mengakses DynamoDB.
- Pilih pada waktunya untuk menjalankan solusi ketika beban kerja tulis pada cluster DB produksi biru Anda akan seringan mungkin. Hindari, misalnya, menjalankan solusi ketika beban massal akan terjadi, atau ketika kemungkinan ada sejumlah besar operasi tulis karena alasan lain.

Menggunakan CloudFormation template untuk menjalankan solusi Neptunus Blue/Green

Anda dapat menggunakan AWS CloudFormation untuk menyebarkan solusi Neptunus Blue/Green . CloudFormationTemplate membuat instans Amazon EC2 di VPC yang sama dengan database Neptunus sumber biru Anda, menginstal solusi di sana, dan menjalankannya. Anda dapat memantau kemajuannya dalam CloudWatch log, seperti yang dijelaskan dalam Memantau kemajuan.

Anda dapat menggunakan tautan ini untuk meninjau template solusi, atau pilih tombol Launch Stack untuk meluncurkannya di CloudFormation konsol:

Lihat	Lihat di Desainer	

Di konsol, pilih AWS wilayah tempat Anda ingin menjalankan solusi dari dropdown di kanan atas jendela.

Atur parameter tumpukan sebagai berikut:

- **`DeploymentID`**— Pengidentifikasi yang unik untuk setiap penyebaran Neptunus Blue/Green .

  Ini digunakan sebagai pengidentifikasi cluster DB hijau, dan sebagai awalan untuk penamaan sumber daya baru yang dibuat selama penerapan.
- **`NeptuneSourceClusterId`**— Pengidentifikasi cluster DB biru yang ingin Anda tingkatkan.
- **`NeptuneTargetClusterVersion:`**— Versi mesin Neptunus yang ingin Anda tingkatkan ke cluster DB biru.

  Ini harus lebih tinggi dari versi mesin cluster DB biru saat ini.
- **`DeploymentMode`**— Menunjukkan apakah ini adalah penerapan baru atau upaya untuk melanjutkan penerapan sebelumnya. Saat Anda menggunakan `DeploymentID` sama dengan penerapan sebelumnya, setel `DeploymentMode` ke`resume`.

  Nilai yang valid adalah: `new` (default), dan`resume`.
- **`GraphQueryType`**— Jenis data grafik untuk database Anda.

  Nilai yang valid adalah: `propertygraph` (default), dan`rdf`.

- **`SubnetId`**— ID subnet dari VPC yang sama tempat cluster DB biru Anda berada. (lihat Menghubungkan ke Cluster DB Neptunus dari instans Amazon EC2 di VPC yang sama).

  Berikan ID subnet publik jika Anda ingin SSH ke instans melalui EC2 Connect.

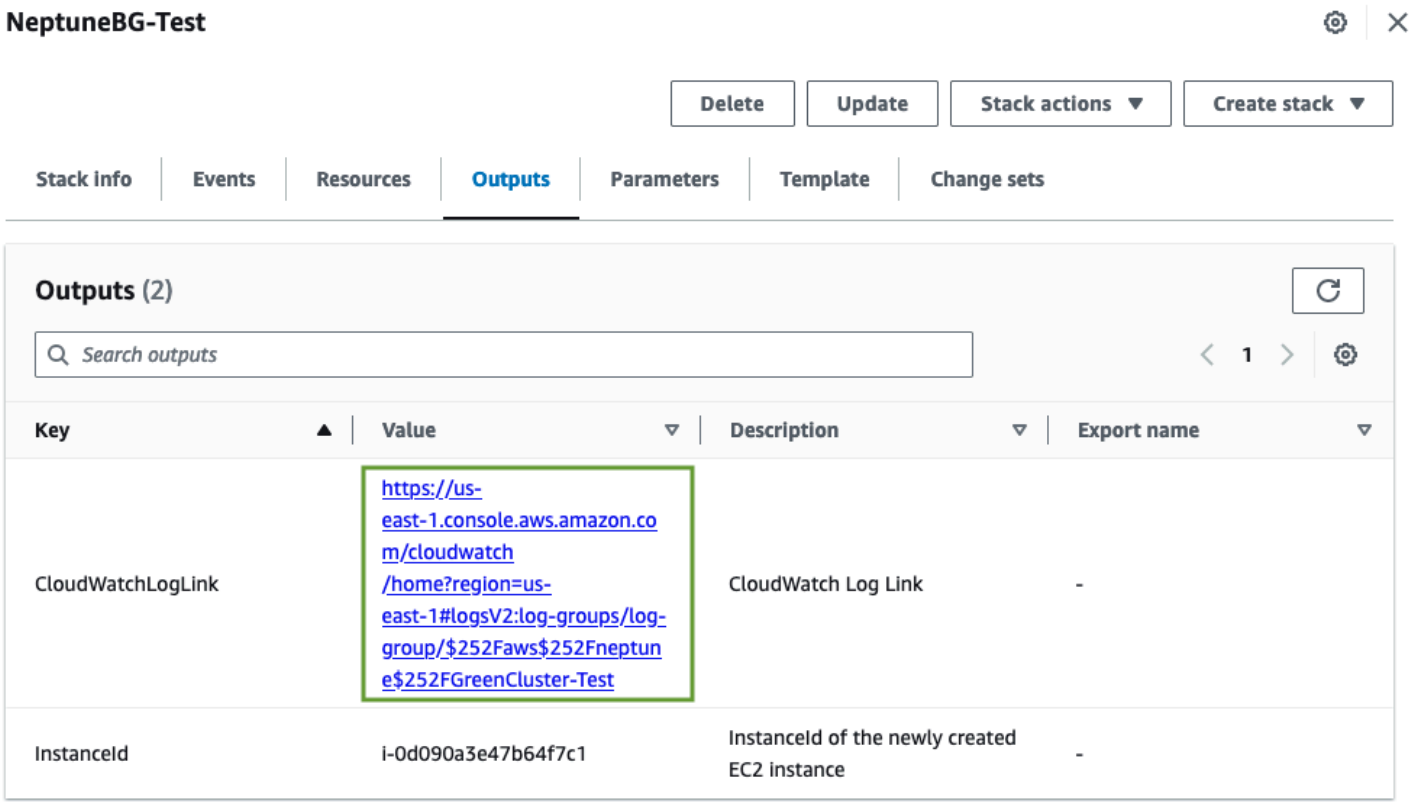
- **`InstanceSecurityGroup`**— Grup keamanan untuk instans Amazon EC2 Anda.

  Grup keamanan harus memiliki akses ke cluster DB biru Anda, dan Anda harus dapat SSH ke instance. Lihat Buat grup keamanan menggunakan konsol VPC.

Tunggu sampai tumpukan selesai. Segera setelah selesai, solusinya dimulai. Anda kemudian dapat memantau proses penyebaran menggunakan CloudWatch log seperti yang dijelaskan di bagian berikutnya.

Memantau kemajuan penyebaran Neptunus Blue/Green

Anda dapat memantau kemajuan solusi Blue/Green Neptunus dengan pergi ke CloudWatch konsol dan melihat log di `/aws/neptune/(Neptune Blue/Green deployment ID)` CloudWatch grup log. Anda dapat menemukan tautan ke CloudWatch log di output CloudFormation tumpukan solusi:



Jika Anda menyediakan subnet publik sebagai parameter tumpukan, Anda juga dapat SSH ke instans Amazon EC2 yang dibuat sebagai bagian dari tumpukan dan merujuk ke log in. `/var/log/cloud-init-output.log`

Log menunjukkan tindakan yang diambil oleh solusi Blue/Green Neptunus, seperti yang ditunjukkan pada tangkapan layar ini:

```
============================================================

Neptune Blue Green Deployment Solution Version: 0.1.06012023

============================================================

Checking whether cluster with id = bg-06-01-14-20-29test-bg1-bgInt already exists.

BlueGreen deployment_mode = new

Didn't find any cluster with id bg-06-01-14-20-29test-bg1-bgInt

Cloned_cluster_id: bg-06-01-14-20-29test-bg1-bgInt

Replication_stack_name: bg-06-01-14-20-29test-bg1-bgInt-replication

DescribeDbClusters response for test-bg1-bgIntegTest-06-01-14-20-29: {'AllocatedStorage': 1,
'AvailabilityZones': ['us-east-1b', 'us-east-1c', 'us-east-1f'], 'BackupRetentionPeriod': 1,
'DBClusterIdentifier': 'test-bg1-bgintegtest-06-01-14-20-29', 'DBClusterParameterGroup': 'green-     -blue-
green-deployment-test-123456789012345-pg-tes710', 'DBSubnetGroup': 'default', 'Status': 'available',
'EarliestRestorableTime': datetime.datetime(2023, 6, 1, 8, 51, 23, 394000, tzinfo=tzlocal()), 'Endpoint':
'test-bg1-bgintegtest-06-01-14-20-29.cluster-critvszpmydm.us-east-1.neptune.amazonaws.com', 'ReaderEndpoint':
'test-bg1-bgintegtest-06-01-14-20-29.cluster-ro-critvszpmydm.us-east-1.neptune.amazonaws.com', 'MultiAZ':
False, 'Engine': 'neptune', 'EngineVersion': '1.2.0.0', 'LatestRestorableTime': datetime.datetime(2023, 6, 1,
8, 51, 23, 394000, tzinfo=tzlocal()), 'Port': 8182, 'MasterUsername': 'admin', 'PreferredBackupWindow':
'06:33-07:03', 'PreferredMaintenanceWindow': 'fri:09:44-fri:10:14', 'ReadReplicaIdentifiers': [],
'DBClusterMembers': [{'DBInstanceIdentifier': 'test-bg1-bgintegtest-06-01-14-20-29i-1', 'IsClusterWriter':
True, 'DBClusterParameterGroupStatus': 'in-sync', 'PromotionTier': 1}], 'VpcSecurityGroups':
```

Pesan log menunjukkan status sinkronisasi antara cluster biru dan hijau:

```
DDB checkpoint {'S': '1'}, {'S': '6'}

DDB Checkpoint: {'checkpointSubSequenceNumber': {'S': '6'}, 'lastUpdateTime': {'N': '1685611142127'},
'leaseOwner': {'S': 'nobody'}, 'checkpoint': {'S': '1'}, 'leaseKey': {'S': 'bg-anl   -234567899-replication'}}

Time difference for last checkpoint and last stream event: 5841351

Stream eventId difference for last replication checkpoint and last stream event on the Source cluster: 0:0

Found region : us-east-1

Cloudwatch Log Url for blue green solution is https://us-east-1.console.aws.amazon.com/cloudwatch
/home?region=us-east-1#logsV2:log-groups/log-group/aws/neptune/bg

Cloudwatch dashboard url for replication is https://console.aws.amazon.com/cloudwatch/home?region=us-
east-1#dashboards:name=neptune-stream-poller-bg-an   -234567899-replication

Replication poller lambda arn is arn:aws:lambda:us-east-1:451235071234:function:bg-an   -234567899-replic-
NeptuneStreamPollerLambd-B6V1ytULgmSP. Look for CW log the poller lambda for more troubleshooting.

Stream Last EventId {'commitNum': 1, 'opNum': 6} on cluster : database-d61852469-t        -experiment.cluster-
critvszpmydm.us-east-1.neptune.amazonaws.com:8182

DDB checkpoint {'S': '1'}, {'S': '6'}

DDB Checkpoint: {'checkpointSubSequenceNumber': {'S': '6'}, 'lastUpdateTime': {'N': '1685611207245'},
'leaseOwner': {'S': 'nobody'}, 'checkpoint': {'S': '1'}, 'leaseKey': {'S': 'bg-ankig-234567899-replication'}}
```

Proses sinkronisasi memeriksa lag replikasi dengan menghitung perbedaan antara aliran terbaru `eventID` pada cluster biru dan pos pemeriksaan replikasi yang ada di tabel pos pemeriksaan DynamoDB yang dibuat oleh tumpukan replikasi. Neptune-to-Neptune Dengan menggunakan pesan-pesan ini, Anda dapat memantau perbedaan replikasi saat ini.

Memotong dari cluster biru produksi ke cluster hijau yang diperbarui

Sebelum mempromosikan cluster hijau ke produksi, pastikan bahwa perbedaan komit antara cluster biru dan hijau adalah nol dan kemudian nonaktifkan semua lalu lintas tulis ke cluster biru. Melanjutkan menulis ke cluster biru sambil mengalihkan titik akhir basis data ke cluster hijau dapat mengakibatkan kerusakan data yang disebabkan oleh penulisan data sebagian ke kedua cluster. Anda mungkin belum perlu menonaktifkan lalu lintas baca.

Jika Anda telah mengaktifkan autentikasi IAM pada klaster sumber (biru), pastikan untuk memperbarui kebijakan IAM yang digunakan dalam aplikasi Anda untuk menunjuk ke klaster hijau (untuk contoh kebijakan semacam itu, lihat kebijakan akses tidak terbatas ini).

Setelah menonaktifkan lalu lintas tulis, tunggu replikasi selesai dan kemudian aktifkan lalu lintas tulis di cluster hijau (tetapi tidak pada cluster biru). Beralih lalu lintas baca dari biru ke cluster hijau juga.

Pembersihan setelah solusi Blue/Green Neptunus selesai

Setelah Anda mempromosikan klaster pementasan (hijau) ke produksi, bersihkan sumber daya yang dibuat oleh solusi Neptunus Blue/Green :

- Hapus instans Amazon EC2 yang dibuat untuk menjalankan solusi.
- Hapus CloudFormation template untuk replikasi berbasis aliran Neptunus yang membuat cluster hijau tetap sinkron dengan cluster biru. Yang utama memiliki nama tumpukan yang Anda berikan sebelumnya, dan satu terdiri dari ID penerapan diikuti oleh “-replikasi”: yaitu,. *(DeploymentID)*-replication

Menghapus CloudFormation template tidak menghapus cluster itu sendiri. Setelah Anda memverifikasi bahwa cluster hijau berfungsi seperti yang diharapkan, Anda dapat mengambil snapshot secara opsional sebelum menghapus cluster biru secara manual.

Praktik terbaik solusi Blue/Green Neptunus

- Sebelum mengalihkan cluster hijau Anda ke produksi, ada baiknya memverifikasi secara menyeluruh bahwa itu berfungsi dengan baik. Periksa konsistensi data dan konfigurasi database. Ada kemungkinan bahwa beberapa versi mesin baru memerlukan peningkatan klien juga. Periksa catatan rilis mesin sebelum Anda meningkatkan. Perlu menguji semua ini dalam pengembangan, pengujian, dan lingkungan pra-produksi sebelum memulai blue/green peningkatan produksi.
- Yang terbaik adalah melakukan peralihan dari server biru ke hijau selama jendela pemeliharaan Anda.
- Untuk memastikan bahwa semuanya berfungsi dengan baik setelah memutakhirkan dan menyinkronkan, ada baiknya menyimpan cluster asli Anda selama beberapa periode waktu sebelum menghapusnya. Itu bisa terbukti berguna jika masalah yang tidak terduga muncul.
- Hindari operasi penulisan berat seperti beban massal saat menjalankan solusi Blue/Green Neptunus, karena dapat menyebabkan kelambatan replikasi yang menyebabkan waktu henti yang signifikan. Idealnya, waktu antara mematikan penulisan ke cluster biru Anda dan menyalakannya untuk cluster hijau Anda hanya beberapa saat.

Memecahkan masalah solusi Neptunus Blue/Green

Informasi berikut menyoroti masalah yang dapat muncul selama proses penerapan Blue/Green solusi, seperti konflik dengan cluster yang ada, kebutuhan untuk mengaktifkan aliran Neptunus,

operasi pemuatan massal yang sedang berlangsung, dan persyaratan kompatibilitas versi. Dengan mengatasi masalah potensial ini, Anda dapat memastikan penerapan solusi Neptunus yang lancar dan sukses. Blue/Green

Kesalahan yang ditimbulkan oleh solusi Neptunus Blue/Green

- **`Cluster with id = (blue_green_deployment_id) already exists`**— Ada cluster yang ada dengan pengenal*`(blue_green_deployment_id)`*.

  Berikan ID penerapan baru atau atur mode penerapan ke `resume` jika cluster dibuat dalam proses Neptunus sebelumnya. Blue/Green
- **`Streams should be enabled on the source Cluster for Blue Green Deployment`**— Aktifkan aliran Neptunus pada cluster biru (sumber).
- **`No Bulkload should be in progress on source cluster: (cluster_id)`**— Solusi Blue/Green Neptunus berakhir jika mengidentifikasi beban curah yang sedang berlangsung.

  Ini untuk memastikan bahwa proses sinkronisasi dapat menangkap penulisan yang sedang dibuat. Hindari atau batalkan pekerjaan pemuatan massal yang sedang berlangsung sebelum memulai solusi Blue/Green Neptunus.
- **`Blue Green deployment requires instances to be in sync with db cluster parameter group`**— Setiap perubahan pada grup parameter cluster harus disinkronkan di seluruh cluster DB. Lihat Grup parameter Amazon Neptunus.
- **`Invalid target engine version for Blue Green Deployment`**— Versi mesin target harus terdaftar sebagai aktifRilis mesin untuk Amazon Neptunus, dan harus lebih tinggi dari pelepasan mesin saat ini dari cluster sumber (biru).

Membuat pengguna IAM dengan izin untuk Neptunus

Untuk mengakses konsol Neptunus untuk membuat dan mengelola cluster DB Neptunus, Anda perlu membuat pengguna IAM dengan semua izin yang diperlukan.

Langkah pertama adalah membuat kebijakan peran terkait layanan untuk Neptunus:

Membuat kebijakan peran terkait layanan untuk Amazon Neptunus

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.
3. Pada halaman Kebijakan, pilih Buat Kebijakan.
4. Pada halaman Buat kebijakan, pilih tab JSON dan salin dalam kebijakan peran terkait layanan berikut:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
      "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"rds.amazonaws.com"
        }
      }
    }
  ]
}
```

5. Pilih Berikutnya: Tag, dan pada halaman Tambahkan tag pilih Berikutnya: Tinjau.
6. Pada halaman Kebijakan ulasan, beri nama kebijakan baru "NeptuneServiceLinked”.

Untuk mengetahui informasi selengkapnya tentang peran terkait layanan, lihat Menggunakan peran terkait layanan untuk Amazon Neptunus.

Buat pengguna IAM baru dengan semua izin yang diperlukan

Selanjutnya, buat pengguna IAM baru dengan kebijakan terkelola yang sesuai yang dilampirkan yang akan memberikan izin yang Anda perlukan, bersama dengan kebijakan peran terkait layanan yang telah Anda buat (di sini bernama): `NeptuneServiceLinked`

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi di sebelah kiri, pilih Pengguna, dan di halaman Pengguna, pilih Tambahkan pengguna.
3. Pada halaman Tambah pengguna, masukkan nama untuk pengguna IAM baru, pilih Kunci akses - Akses program untuk jenis AWS kredensi, dan pilih Berikutnya: Izin.
4. Pada halaman Setel izin, di kotak Kebijakan filter, ketik “Neptunus”. Sekarang pilih yang berikut ini dari kebijakan yang tercantum:

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked(dengan asumsi itulah yang Anda beri nama kebijakan peran terkait layanan yang Anda buat sebelumnya).
5. Ketik berikutnya “VPC” di kotak Kebijakan filter sebagai pengganti “Neptunus”. Pilih Amazon VPCFull Access dari kebijakan yang tercantum.
6. Pilih Berikutnya: Tag, dan di halaman Tambahkan tag, pilih Berikutnya: Tinjau.
7. Di halaman Tinjauan, periksa apakah semua kebijakan berikut sekarang dilampirkan ke pengguna baru Anda:

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked
   - VPCFullAkses Amazon

   Kemudian, pilih Buat Pengguna.
8. Terakhir, unduh dan simpan ID kunci akses pengguna baru dan kunci akses rahasia.

Untuk beroperasi di layanan lain seperti Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3), Anda perlu menambahkan lebih banyak izin dan hubungan kepercayaan.

Grup parameter Amazon Neptunus

Anda mengelola konfigurasi database Anda di Amazon Neptunus dengan menggunakan parameter dalam grup parameter. Grup parameter bertindak sebagai wadah untuk nilai konfigurasi mesin yang diterapkan ke satu atau lebih instance DB.

Ada dua jenis kelompok parameter, yaitu kelompok parameter cluster DB dan kelompok parameter DB:

- Grup parameter DB berlaku di tingkat instance dan umumnya dikaitkan dengan pengaturan untuk mesin grafik Neptune, seperti parameter `neptune_query_timeout`.
- Grup parameter klaster DB berlaku untuk setiap instans dalam klaster dan umumnya memiliki pengaturan yang lebih luas. Setiap klaster Neptune dikaitkan dengan grup parameter klaster DB. Setiap instans DB dalam cluster itu mewarisi nilai konfigurasi engine yang terdapat dalam grup parameter cluster DB.

Setiap nilai konfigurasi apa pun yang Anda ubah di grup parameter klaster DB akan menimpa nilai default di grup parameter DB. Jika Anda mengedit nilai yang sesuai dalam grup parameter DB, nilai-nilai tersebut mengesampingkan pengaturan di grup parameter cluster DB.

Grup parameter DB default digunakan jika Anda membuat instans DB tanpa menentukan grup parameter DB kustom. Anda tidak dapat mengubah pengaturan parameter grup parameter DB default. Sebagai gantinya, untuk mengubah pengaturan parameter default Anda harus membuat grup parameter DB baru. Tidak semua parameter mesin DB dapat diubah dalam grup parameter DB yang Anda buat.

Grup parameter dibuat dalam keluarga yang kompatibel dengan versi mesin Neptunus tertentu. Saat Anda meningkatkan ke versi mesin mayor atau minor baru, Anda mungkin perlu membuat ulang grup parameter kustom Anda menggunakan keluarga grup parameter yang sesuai untuk versi tersebut.

Penamaan keluarga grup parameter mengikuti pola`neptune`*X*`.`*Y*, di mana *X*`.`*Y* cocok dengan versi mesin. Contoh:

- `neptune1`— untuk versi mesin sebelum 1.2.0.0
- `neptune1.2`- untuk versi mesin 1.2.x
- `neptune1.3`- untuk versi mesin 1.3.x
- `neptune1.4`- untuk versi mesin 1.4.x

Saat memutakhirkan cluster Neptunus Anda, periksa catatan rilis untuk versi mesin target Anda untuk menentukan apakah keluarga grup parameter baru diperlukan. Jika demikian, Anda harus membuat ulang semua grup parameter kustom dalam keluarga baru sebelum memutakhirkan.

Beberapa parameter Neptunus bersifat statis, dan yang lainnya dinamis. Perbedaannya adalah sebagai berikut:

Parameter statis

- Parameter statis adalah parameter yang berlaku hanya setelah instance DB di-boot ulang. Dengan kata lain, ketika Anda mengubah parameter statis dan menyimpan grup parameter DB instance, Anda harus secara manual me-reboot instance DB agar perubahan parameter diterapkan. Saat ini, semua parameter tingkat instans Neptunus (dalam grup parameter DB daripada grup parameter cluster DB) bersifat statis.
- Saat Anda mengubah parameter statis tingkat cluster dan menyimpan grup parameter cluster DB, perubahan parameter akan berlaku setelah Anda me-reboot setiap instans DB di cluster secara manual.

Parameter dinamis

- Parameter dinamis adalah parameter yang berlaku segera setelah parameter diperbarui dalam grup parameternya. Dengan kata lain, tidak perlu me-reboot instance DB setelah memperbarui parameter dinamis agar perubahan parameter diterapkan.
- Harapkan beberapa penundaan kecil untuk perubahan parameter cluster dinamis diterapkan di semua instance DB.
- Nilai parameter dinamis yang diperbarui tidak diterapkan pada permintaan yang sedang berjalan, tetapi hanya untuk permintaan yang dikirimkan setelah perubahan terjadi.
- Saat Anda mengubah parameter tingkat cluster dinamis, secara default perubahan parameter diterapkan ke cluster DB Anda segera, tanpa memerlukan reboot apa pun. Untuk menunda perubahan parameter sampai setelah instance DB di cluster di-boot ulang, Anda dapat menggunakan AWS CLI untuk mengatur to untuk perubahan parameter. `ApplyMethod` `pending-reboot`

Saat ini semua parameter statis kecuali untuk parameter cluster baru berikut:

- `neptune_enable_slow_query_log`(tingkat cluster)
- `neptune_slow_query_log_threshold`(tingkat cluster)

Berikut ini beberapa poin penting yang harus Anda ketahui tentang bekerja dengan parameter dalam grup parameter DB:

- Pengaturan parameter yang tidak tepat dalam grup parameter DB dapat memiliki efek merugikan yang tidak diinginkan, termasuk penurunan kinerja dan ketidakstabilan sistem. Selalu berhati-hati saat memodifikasi parameter basis data dan membuat cadangan data sebelum memodifikasi grup parameter DB. Coba perubahan pengaturan grup parameter pada instans DB uji sebelum menerapkan perubahan tersebut ke instans DB produksi.
- Saat Anda mengubah grup parameter DB yang terkait dengan instans DB, Anda harus me-reboot instance secara manual sebelum grup parameter DB baru digunakan oleh instans DB.

> **Note**
>
> SebelumnyaRilis: 1.2.0.0 (2022-07-21), semua instance baca-replika dalam cluster DB secara otomatis di-boot ulang setiap kali instance primer (penulis) dimulai ulang. Dari Rilis: 1.2.0.0 (2022-07-21) maju, memulai ulang instance utama tidak menyebabkan instance replika apa pun dimulai ulang. Ini berarti bahwa jika Anda mengubah parameter tingkat cluster, Anda harus memulai ulang setiap instance secara terpisah untuk mengambil perubahan parameter.

Mengedit Grup Parameter Klaster DB atau Group Parameter DB

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Pilih Grup parameter di panel navigasi.
3. Pilih link Nama untuk grup parameter DB yang ingin Anda edit.

   (Opsional) Pilih Buat grup parameter untuk membuat grup parameter klaster baru dan membuat grup baru. Lalu pilih Nama grup parameter baru.

   > **Important**
   >
   > Langkah ini adalah wajib Jika Anda hanya memiliki grup parameter klaster DB default karena grup parameter klaster DB tidak dapat diubah.

4. Cari parameter dan klik pada bidang Nilai di sebelah kolom Nama.

5. Masukkan nilai yang diizinkan dan pilih centang di samping bidang nilai.
6. Pilih Simpan perubahan.
7. Reboot setiap instans DB di cluster Neptunus jika Anda mengubah parameter cluster DB, atau satu atau lebih instance spesifik jika Anda mengubah parameter instans DB.

Membuat grup parameter DB atau grup parameter cluster DB

Anda dapat dengan mudah menggunakan konsol Neptunus untuk membuat grup parameter baru:

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Pilih Grup Parameter Di panel navigasi kiri.
3. Pilih Buat grup parameter DB.

   Halaman Buat grup parameter DB akan muncul.
4. Dalam daftar keluarga grup Parameter, pilih keluarga yang cocok dengan versi mesin Neptunus target Anda (misalnya, neptune1.2, neptune1.3, atau neptune1.4).
5. Di daftar Jenis, pilih Grup parameter DB atau Grup Parameter Klaster DB.
6. Di kotak Nama grup, masukkan nama grup parameter DB baru.
7. Di kotak Deskripsi, masukkan deskripsi untuk grup parameter DB baru.
8. Pilih Buat.

Anda juga dapat membuat grup parameter baru menggunakan AWS CLI:

```
aws neptune create-db-parameter-group \
  --db-parameter-group-name (a name for the new DB parameter group) \
  --db-parameter-group-family (the family matching your engine version, such as
 neptune1.2, neptune1.3, or neptune1.4) \
  --description (a description for the new DB parameter group)
```

Parameter Amazon Neptunus

Anda mengelola konfigurasi database Anda di Amazon Neptunus dengan menggunakan parameter dalam grup parameter. Parameter berikut tersedia untuk mengonfigurasi database Neptunus Anda:

Parameter tingkat klaster

- neptune_enable_audit_log
- neptune_enable_slow_query_log
- neptune_slow_query_log_threshold
- neptune_lab_mode
- neptune_query_timeout
- neptune_stream
- neptune_streams_expiry_days
- neptune_lookup_cache
- neptune_autoscaling_config
- neptune_ml_iam_role
- neptune_ml_titik akhir
- neptune_enable_inline_server_generated_edge_id

Parameter tingkat instans

- neptune_dfe_query_engine
- neptune_query_timeout
- neptune_result_cache
- UndoLogPurgeConfig

Parameter usang

- neptune_enforce_ssl

`neptune_enable_audit_log`(parameter tingkat cluster)

Parameter ini mengaktifkan pencatatan audit untuk Neptunus.

Nilai yang diizinkan adalah `0` (dinonaktifkan) dan `1` (diaktifkan). Nilai default-nya adalah `0`.

Parameter ini statis, artinya perubahannya tidak berlaku pada instance apa pun sampai di-boot ulang.

Anda dapat mempublikasikan log audit ke Amazon CloudWatch, seperti yang dijelaskan dalamMenggunakan CLI untuk menerbitkan log audit Neptunus ke Log CloudWatch.

`neptune_enable_slow_query_log`(parameter tingkat cluster)

Gunakan parameter ini untuk mengaktifkan atau menonaktifkan fitur logging kueri lambat Neptunus.

Ini adalah parameter dinamis, yang berarti bahwa mengubah nilainya tidak memerlukan atau menyebabkan restart cluster DB Anda.

Nilai yang diizinkan adalah:

- **`info`**— Mengaktifkan pencatatan kueri lambat dan mencatat atribut yang dipilih yang mungkin berkontribusi pada kinerja yang lambat.
- **`debug`**— Mengaktifkan pencatatan kueri lambat dan mencatat semua atribut yang tersedia dari kueri yang dijalankan.
- **`disabled`**— Menonaktifkan pencatatan kueri lambat.

Nilai default-nya adalah `disabled`.

Anda dapat mempublikasikan log kueri lambat ke Amazon CloudWatch, seperti yang dijelaskan dalam. Menggunakan CLI untuk menerbitkan log kueri lambat Neptunus ke Log CloudWatch

`neptune_slow_query_log_threshold`(parameter tingkat cluster)

Parameter ini menentukan ambang waktu eksekusi, dalam milidetik, setelah itu kueri dianggap sebagai kueri lambat. Jika pencatatan kueri lambat diaktifkan, kueri yang berjalan lebih lama dari ambang batas ini akan dicatat bersama dengan beberapa atributnya.

Nilai default adalah 5000 milidetik (5 detik).

Ini adalah parameter dinamis, yang berarti bahwa mengubah nilainya tidak memerlukan atau menyebabkan restart cluster DB Anda.

`neptune_lab_mode`(parameter tingkat cluster)

Saat diatur, parameter ini memungkinkan fitur eksperimental spesifik Neptunus. Lihat Mode Lab Neptune untuk fitur eksperimental yang tersedia saat ini.

Parameter ini statis, artinya perubahannya tidak berlaku pada instance apa pun sampai di-boot ulang.

Untuk mengaktifkan atau menonaktifkan fitur eksperimental, sertakan *`(feature name)`* `=enabled` atau *`(feature name)`* `=disabled` dalam parameter ini. Anda dapat mengaktifkan atau menonaktifkan beberapa fitur dengan memisahkannya dengan koma, seperti ini:

`(feature #1 name)``=enabled,` *`(feature #2 name)`*`=enabled`

Fitur mode lab biasanya dinonaktifkan secara default. Pengecualian adalah `DFEQueryEngine` fitur, yang diaktifkan secara default untuk digunakan dengan petunjuk kueri (`DFEQueryEngine=viaQueryHint`) dimulai pada rilis mesin Neptunus 1.0.5.0. Dimulai dengan rilis mesin Neptunus 1.1.1.0, mesin DFE tidak lagi dalam mode lab, dan sekarang dikontrol menggunakan neptune_dfe_query_engine parameter instance dalam grup parameter DB instans.

`neptune_query_timeout`(parameter tingkat cluster)

Menentukan durasi batas waktu tertentu untuk kueri grafik, dalam milidetik.

Nilai yang diizinkan berkisar dari `10` ke `2,147,483,647` (2^{31} - 1). Nilai defaultnya adalah `120,000` (2 menit).

Parameter ini statis, artinya perubahannya tidak berlaku pada instance apa pun sampai di-boot ulang.

Ketika beberapa pengaturan batas waktu dikonfigurasi (tingkat klaster, tingkat instance, dan per-kueri), tabel berikut menunjukkan nilai batas waktu mana yang diutamakan:

Kluster PG	Contoh PG	Petunjuk Kueri	Hasil
Default	Default	none	Kluster
Khusus	Default	none	Kluster
Khusus	Khusus	none	Instans
Default	Khusus	none	Instans

Kluster PG	Contoh PG	Petunjuk Kueri	Hasil
Setiap	Setiap	terendah	Kueri
Default	Khusus	tidak terendah	Instans
Khusus	Default	tidak terendah	Kluster
Khusus	Khusus	tidak terendah	Instans

> **Note**
>
> Dimungkinkan untuk mengeluarkan biaya tak terduga jika Anda menetapkan nilai batas waktu kueri terlalu tinggi, terutama pada instance tanpa server. Tanpa pengaturan batas waktu yang wajar, Anda mungkin secara tidak sengaja mengeluarkan kueri yang terus berjalan lebih lama dari yang Anda harapkan, menimbulkan biaya yang tidak pernah Anda antisipasi. Hal ini terutama berlaku pada instance tanpa server yang dapat meningkatkan skala hingga jenis instance yang besar dan mahal saat menjalankan kueri.
> Anda dapat menghindari pengeluaran tak terduga semacam ini dengan menggunakan nilai batas waktu kueri yang mengakomodasi sebagian besar kueri Anda dan hanya menyebabkan yang berjalan lama secara tak terduga habis.

`neptune_streams`(parameter tingkat cluster)

Mengaktifkan atau menonaktifkanAliran Neptunus.

Parameter ini statis, artinya perubahannya tidak berlaku pada instance apa pun sampai di-boot ulang.

Nilai yang diizinkan adalah `0` (dinonaktifkan, yang merupakan default), dan`1` (diaktifkan).

`neptune_streams_expiry_days`(parameter tingkat cluster)

Menentukan berapa hari berlalu sebelum server menghapus catatan aliran.

Nilai yang diizinkan adalah dari 1 ke`90`, inklusif. Nilai default-nya 7.

Parameter ini diperkenalkan di versi mesin 1.2.0.0.

Parameter ini statis, artinya perubahannya tidak berlaku pada instance apa pun sampai di-boot ulang.

`neptune_lookup_cache`(parameter tingkat cluster)

Menonaktifkan atau mengaktifkan kembali cache pencarian Neptunus. pada `R5d` contoh.

Parameter ini statis, artinya perubahannya tidak berlaku pada instance apa pun sampai di-boot ulang.

Nilai yang diizinkan adalah `1` (diaktifkan) dan `0` (dinonaktifkan). Nilai defaultnya adalah`0`, tetapi setiap kali sebuah `R5d` instance dibuat di cluster DB, `neptune_lookup_cache` parameter secara otomatis diatur ke `1` dan cache pencarian dibuat pada instance itu.

`neptune_autoscaling_config`(parameter tingkat cluster)

Menetapkan parameter konfigurasi untuk instance read-replica yang dibuat dan dikelola oleh Neptune auto-scaling.

Parameter ini statis, artinya perubahannya tidak berlaku pada instance apa pun sampai di-boot ulang.

Menggunakan string JSON yang Anda tetapkan sebagai nilai `neptune_autoscaling_config` parameter, Anda dapat menentukan:

- Jenis instans yang digunakan auto-scaling Neptunus untuk semua instance read-replica baru yang dibuatnya.
- Jendela pemeliharaan ditetapkan untuk replika baca tersebut.
- Tag yang akan dikaitkan dengan semua replika baca baru.

String JSON memiliki struktur seperti ini:

```
"{
  \"tags\": [
    { \"key\" : \"reader tag-0 key\", \"value\" : \"reader tag-0 value\" },
    { \"key\" : \"reader tag-1 key\", \"value\" : \"reader tag-1 value\" },
  ],
  \"maintenanceWindow\" : \"wed:12:03-wed:12:33\",
  \"dbInstanceClass\" : \"db.r5.xlarge\"
}"
```

Perhatikan bahwa tanda kutip dalam string semua harus lolos dengan karakter garis miring terbalik (). \

Salah satu dari tiga pengaturan konfigurasi yang tidak ditentukan dalam `neptune_autoscaling_config` parameter disalin dari konfigurasi instance penulis utama cluster DB.

`neptune_ml_iam_role`(parameter tingkat cluster)

Menentukan peran IAM ARN yang digunakan dalam Neptunus ML. Nilai dapat berupa ARN peran IAM yang valid.

Parameter ini statis, artinya perubahannya tidak berlaku pada instance apa pun sampai di-boot ulang.

Anda dapat menentukan ARN peran IAM default untuk pembelajaran mesin pada grafik.

`neptune_ml_endpoint`(parameter tingkat cluster)

Menentukan endpoint yang digunakan untuk Neptunus ML. Nilainya bisa berupa nama titik akhir SageMaker AI yang valid.

Parameter ini statis, artinya perubahannya tidak berlaku pada instance apa pun sampai di-boot ulang.

Anda dapat menentukan titik akhir SageMaker AI default untuk pembelajaran mesin pada grafik.

`neptune_enable_inline_server_generated_edge_id`(parameter tingkat cluster)

Aktifkan atau nonaktifkan fitur ID Edge yang dihasilkan server sebaris Neptunus.

Parameter ini statis, artinya perubahannya tidak berlaku pada instance apa pun sampai di-boot ulang.

Nilai yang diizinkan adalah `1` (diaktifkan), dan `0` (dinonaktifkan). Nilai default-nya adalah `0`.

`neptune_dfe_query_engine`(parameter tingkat contoh)

Dimulai dengan rilis mesin Neptunus 1.1.1.0, parameter instans DB ini digunakan untuk mengontrol bagaimana mesin kueri DFE digunakan. Nilai yang diizinkan adalah sebagai berikut:

Parameter ini statis, artinya perubahannya tidak berlaku pada instance apa pun sampai di-boot ulang.

- **`enabled`**— Menyebabkan mesin DFE digunakan sedapat mungkin, kecuali di mana petunjuk `useDFE` kueri ada dan disetel ke. `false`
- **`viaQueryHint`**(default) - Menyebabkan mesin DFE hanya digunakan untuk kueri yang secara eksplisit menyertakan petunjuk kueri yang disetel ke`useDFE`. `true`

Jika parameter ini belum ditetapkan secara eksplisit, nilai default,`viaQueryHint`, digunakan ketika instance dimulai.

> **Note**
>
> Semua kueri OpenCypher dijalankan oleh mesin DFE terlepas dari bagaimana parameter ini diatur.

Sebelum merilis 1.1.1.0, ini adalah parameter lab-mode daripada parameter instans DB.

`neptune_query_timeout`(parameter tingkat contoh)

Parameter instans DB ini menentukan durasi batas waktu untuk kueri grafik, dalam milidetik, untuk satu contoh.

Parameter ini statis, artinya perubahannya tidak berlaku pada instance apa pun sampai di-boot ulang.

Nilai yang diizinkan berkisar dari `10` ke `2,147,483,647` (2^{31} - 1). Nilai defaultnya adalah `120,000` (2 menit).

> **Note**
>
> Dimungkinkan untuk mengeluarkan biaya tak terduga jika Anda menetapkan nilai batas waktu kueri terlalu tinggi, terutama pada instance tanpa server. Tanpa pengaturan batas waktu yang wajar, Anda mungkin secara tidak sengaja mengeluarkan kueri yang terus berjalan lebih lama dari yang Anda harapkan, menimbulkan biaya yang tidak pernah Anda antisipasi. Hal ini terutama berlaku pada instance tanpa server yang dapat meningkatkan skala hingga jenis instance yang besar dan mahal saat menjalankan kueri.
> Anda dapat menghindari pengeluaran tak terduga semacam ini dengan menggunakan nilai batas waktu kueri yang mengakomodasi sebagian besar kueri Anda dan hanya menyebabkan yang berjalan lama secara tak terduga habis.

`neptune_result_cache`(parameter tingkat contoh)

`neptune_result_cache`— Parameter instans DB ini memungkinkan atau menonaktifkanHasil kueri cache.

Parameter ini statis, artinya perubahannya tidak berlaku pada instance apa pun sampai di-boot ulang.

Nilai yang diizinkan adalah`0`, (dinonaktifkan, yang merupakan default), dan 1 (diaktifkan).

`UndoLogPurgeConfig`(parameter tingkat contoh)

`UndoLogPurgeConfig`— Gunakan parameter ini untuk mengaktifkan atau menonaktifkan UndoLog pembersihan agresif di Neptunus.

Nilai yang diizinkan adalah`default`, yang menggunakan jumlah standar utas untuk membatalkan pembersihan log, dan`aggressive`, yang menggunakan peningkatan jumlah utas untuk mempercepat pembersihan log pembatalan. Ketika `agressive` opsi dipilih, Anda dapat mengharapkan untuk mengamati nilai yang lebih besar untuk metrik. `NumUndoPagesPurged`

`neptune_enforce_ssl`(Parameter tingkat cluster usang)

(Usang) Dulu ada wilayah yang mengizinkan koneksi HTTP ke Neptunus, dan parameter ini digunakan untuk memaksa semua koneksi menggunakan HTTPS saat disetel ke 1. Parameter ini tidak lagi relevan, namun, karena Neptune sekarang hanya menerima koneksi HTTPS di semua wilayah.

Meluncurkan cluster DB Neptunus menggunakan Konsol Manajemen AWS

Cara termudah untuk meluncurkan cluster DB Neptunus baru adalah dengan menggunakan template CloudFormation yang membuat semua sumber daya yang diperlukan untuk Anda, seperti yang dijelaskan dalam. Buat cluster Neptunus

Jika mau, Anda juga dapat menggunakan konsol Neptunus untuk meluncurkan cluster DB baru secara manual, seperti yang dijelaskan di sini.

> **Note**
>
> Sebelum Anda dapat mengakses konsol Neptunus untuk membuat cluster Neptunus, Anda harus memiliki pengguna dengan izin yang cukup. Jika pengguna Anda saat ini tidak memiliki izin ini maka Anda dapat membuat pengguna IAM dengan izin yang diperlukan untuk melakukannya, seperti yang dijelaskan di. Membuat pengguna IAM dengan izin untuk Neptunus

Setelah Anda memverifikasi bahwa pengguna Anda memiliki izin yang benar, atau Anda telah membuat pengguna dengan izin yang benar, masuk ke Konsol Manajemen AWS sebagai pengguna IAM tersebut dan ikuti langkah-langkah di bawah ini untuk membuat cluster DB baru:

Meluncurkan klaster DB Neptune menggunakan konsol

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Arahkan ke halaman Clusters di bawah Databases dan pilih Create database, yang membuka halaman Create database.
3. Di bawah Pengaturan, masukkan nama untuk cluster DB baru Anda atau terima nama default yang disediakan di sana. Nama ini digunakan di alamat titik akhir instance, dan harus memenuhi batasan berikut:

   - Pengidentifikasi ini harus berisi 1 hingga 63 karakter alfanumerik atau tanda hubung.
   - Karakter pertamanya harus berupa huruf.
   - Pengidentifikasi ini tidak boleh diakhiri dengan tanda hubung atau mengandung dua tanda hubung berturut-turut.

- Itu harus unik di semua instans DB di AWS akun Anda di AWS Wilayah tertentu.

4. Di bawah Template, pilih Produksi atau Pengembangan dan Pengujian.
5. Di bawah ukuran instans DB, pilih ukuran instans. Ini akan menentukan kapasitas pemrosesan dan memori dari instance penulisan utama cluster DB baru Anda.

   Jika Anda memilih template Produksi, Anda hanya dapat memilih dari antara kelas yang dioptimalkan memori yang tersedia yang terdaftar, tetapi jika Anda memilih Pengembangan dan pengujian, Anda juga dapat memilih dari antara kelas burstable yang lebih ekonomis (lihat Instans Burstable T3 untuk diskusi tentang kelas burstable).

   > **Note**
   >
   > Neptunus tidak lagi R4 mendukung tipe instans.

6. Di bawah Ketersediaan dan daya tahan, Anda dapat memilih apakah akan mengaktifkan penyebaran multi-availability-zone (Multi-AZ) atau tidak. Template produksi memungkinkan penerapan Multi-AZ secara default, sedangkan template pengembangan dan pengujian tidak. Jika penerapan multi-AZ diaktifkan, Neptunus akan menemukan instance read-replica yang Anda buat di zona ketersediaan yang berbeda () untuk meningkatkan ketersediaan. AZs
7. Di bawah Konektivitas, pilih virtual private cloud (VPC) yang akan meng-host cluster DB baru Anda dari salah satu pilihan yang tersedia. Di sini Anda dapat memilih Buat VPC baru jika Anda ingin Neptunus membuat VPC untuk Anda. Anda harus membuat instans Amazon EC2 di VPC yang sama ini untuk mengakses instans Neptunus (untuk informasi selengkapnya, lihat). Mengamankan database Amazon Neptunus Anda dengan Amazon VPC Perhatikan bahwa Anda tidak dapat mengubah VPC setelah cluster DB dibuat.

   Jika perlu, Anda dapat mengonfigurasi konektivitas lebih lanjut untuk klaster Anda di bawah Konfigurasi konektivitas tambahan:

   a. Di bawah grup Subnet, Anda dapat memilih grup subnet Neptunus DB yang akan digunakan untuk cluster DB baru. Jika VPC Anda belum memiliki grup subnet, Neptunus membuat grup subnet DB untuk Anda (lihat). Mengamankan database Amazon Neptunus Anda dengan Amazon VPC
   b. Di bawah grup keamanan VPC, pilih satu atau beberapa grup keamanan VPC yang ada untuk mengamankan akses jaringan ke kluster DB baru, atau pilih Buat baru jika Anda ingin Neptunus membuatnya untuk Anda, lalu berikan nama untuk grup keamanan VPC baru (lihat). Buat grup keamanan menggunakan konsol VPC

c. Di bawah port Database, masukkan TCP/IP port yang akan digunakan database untuk koneksi aplikasi. Neptunus menggunakan 8182 nomor port sebagai default.

8. Di bawah konfigurasi Notebook, pilih Buat buku catatan jika Anda ingin Neptunus membuat notebook Jupyter untuk Anda di meja kerja Neptunus (lihat dan). Menggunakan Amazon Neptunus dengan notebook grafik Menggunakan workbench Neptune untuk meng-host notebook Neptune Anda kemudian dapat memilih bagaimana notebook baru harus dikonfigurasi:

   a. Di bawah jenis instans Notebook, pilih salah satu kelas instans yang tersedia untuk notebook Anda.

   b. Di bawah nama Notebook, masukkan nama untuk buku catatan Anda.

   c. Jika mau, Anda juga dapat memasukkan deskripsi notebook di bawah Deskripsi - opsional.

   d. Di bawah nama peran IAM, pilih agar Neptunus membuat peran IAM untuk buku catatan, dan masukkan nama untuk peran baru, atau pilih untuk memilih peran IAM yang ada dari antara peran yang tersedia.

   e. Terakhir, pilih apakah notebook Anda terhubung ke internet secara langsung atau melalui Amazon SageMaker AI atau melalui VPC dengan gateway NAT. Lihat Connect Instance Notebook ke Resources di VPC untuk informasi selengkapnya.

9. Di bawah Tag, Anda dapat mengaitkan hingga 50 tag dengan cluster DB baru Anda.

10. Di bawah Konfigurasi tambahan, ada lebih banyak pengaturan yang dapat Anda buat untuk cluster DB baru Anda (dalam banyak kasus, Anda dapat melewatinya dan menerima nilai default untuk saat ini):

Opsi	Apa yang dapat Anda lakukan
Pengidentifikasi instans DB	Anda dapat memberikan nama untuk instance penulis cluster. Jika tidak, pengenal default berdasarkan nama cluster digunakan . Jika Anda melakukannya, tentukan nama yang unik untuk semua instans DB yang dimiliki oleh AWS akun Anda di wilayah saat ini. Pengidentifikasi instans DB tidak peka huruf besar/kecil, tetapi disimpan sebagai semua huruf kecil.

Opsi	Apa yang dapat Anda lakukan
Grup parameter klaster basis data	Pilih grup parameter cluster DB untuk menentukan konfigurasi default untuk semua instance DB di cluster. Kecuali Anda memilih sebaliknya, Neptunus menggunakan grup parameter cluster DB default. Untuk informasi selengkapnya tentang grup parameter, lihat Grup parameter Amazon Neptunus.
Grup parameter DB	Pilih grup parameter DB untuk menentukan konfigurasi instans DB utama di cluster. Kecuali Anda memilih sebaliknya, Neptunus menggunakan grup parameter default. Untuk informasi selengkapnya tentang grup parameter, lihat Grup parameter.
Autentikasi IAM DB	Jika Anda memeriksa Aktifkan autentikasi IAM DB, semua akses ke database Anda akan diautentikasi menggunakan AWS Identity and Access Management (IAM). **Important** Ini mengharuskan Anda menandatangani semua permintaan dengan penandatanganan AWS Signature Version 4. Untuk informasi selengkapnya, lihat Mengautentikasi database Amazon Neptunus Anda dengan AWS Identity and Access Management.

Opsi	Apa yang dapat Anda lakukan
Prioritas failover	Pilih `No preference` atau tingkat prioritas untuk failover. Jika Anda memilih tier dan ada pertengkaran di dalamnya, replika yang ukurannya sama dengan instance utama dipilih.
Periode retensi cadangan	Pilih lamanya waktu, dari 1 hingga 35 hari, bahwa Neptunus harus menyimpan cadangan otomatis instans DB ini. Anda hanya dapat melakukan point-in-time pemulihan (PITR) ke waktu dalam periode retensi cadangan.
Salin tag ke cuplikan	(Diaktifkan secara default) Opsi ini menyebabkan semua tag yang terkait dengan cluster DB Anda disalin ke snapshot apa pun darinya.
Aktifkan enkripsi	(Diaktifkan secara default) Opsi ini menyebabkan data di cluster DB Anda dienkripsi saat istirahat. Jika Anda melakukannya, pilih kunci master yang digunakan untuk melindungi kunci yang digunakan untuk mengenkripsi volume database ini. Anda dapat memilih `aws/rds` kunci default, atau memilih dari kunci master di akun Anda, atau memasukkan ARN kunci dari akun yang berbeda. Anda dapat membuat kunci enkripsi master baru pada tab Encryption Keys pada konsol IAM. Untuk informasi selengkapnya, lihat Mengenkripsi data saat istirahat di database Amazon Neptunus Anda.

Opsi	Apa yang dapat Anda lakukan
Log audit	Periksa ini jika Anda ingin log audit dari cluster DB Anda dipublikasikan ke CloudWatch Log.
Aktifkan peningkatan versi minor otomatis	(Diaktifkan secara default) Opsi ini menyebabkan cluster DB Anda ditingkat kan secara otomatis ke versi mesin minor baru setelah dirilis. Peningkatan otomatis terjadi selama periode pemeliharaan untuk basis data. Lihat Menggunakan AutoMinor VersionUpgrade.
Jendela pemeliharaan	Anda dapat memilih periode tertentu di mana Anda ingin modifikasi tertunda pada cluster DB Anda terjadi, seperti perubahan ke kelas instans DB atau patch mesin otomatis. Setiap operasi pemeliharaan tersebut dimulai dan diselesaikan dalam periode yang dipilih. Jika Anda tidak memilih periode, Neptunus menetapkan periode pemeliharaan secara sewenang-wenang.
Aktifkan perlindungan penghapusan	(Diaktifkan secara default) Perlindungan penghapusan memblokir cluster DB Anda agar tidak dihapus. Anda harus menonakti fkannya secara eksplisit untuk menghapus cluster DB.

11. Pilih Buat database untuk meluncurkan cluster DB Neptunus baru Anda dan instans utamanya.

    Pada konsol Amazon Neptune, klaster DB baru muncul dalam daftar basis data. Klaster DB memiliki status Membuat hingga klaster DB tersebut dibuat dan siap digunakan. Saat status berubah menjadi Tersedia, Anda dapat terhubung ke instans primer untuk klaster DB Anda. Bergantung pada kelas instans DB dan penyimpanan yang dialokasikan, itu dapat memerlukan beberapa menit agar instans baru tersedia.

Untuk melihat klaster yang baru dibuat, pilih tampilan Basis data di Neptune konsol.

> **Note**
>
> Jika Anda menghapus semua instans DB Neptunus di cluster DB menggunakan Konsol Manajemen AWS, konsol secara otomatis menghapus cluster DB itu sendiri. Jika Anda menggunakan AWS CLI atau SDK, Anda harus menghapus cluster DB secara manual setelah Anda menghapus instance terakhirnya.

Catat nilai endpoint Cluster. Anda perlu ini untuk terhubung ke klaster DB Neptune Anda.

Menghentikan dan memulai cluster DB Amazon Neptunus

Menghentikan dan memulai klaster Amazon Neptune membantu Anda mengelola biaya untuk pengembangan dan pengujian lingkungan. Anda dapat menghentikan sementara semua instans DB di klaster Anda, bukan mengatur dan merombak semua instans DB setiap kali Anda menggunakan klaster.

Topik

- Ikhtisar menghentikan dan memulai cluster DB Neptunus
- Menghentikan cluster DB Neptunus
- Memulai cluster DB Neptunus yang berhenti

Ikhtisar menghentikan dan memulai cluster DB Neptunus

Selama periode ketika Anda tidak memerlukan klaster Neptune, Anda dapat menghentikan semua instans dalam klaster itu sekaligus. Anda dapat memulai klaster lagi kapan pun diperlukan. Memulai dan menghentikan akan menyederhanakan proses penyiapan dan perombakan pada klaster yang digunakan untuk pengembangan, pengujian, atau aktivitas serupa yang tidak memerlukan ketersediaan yang berkelanjutan. Anda dapat mencapai ini Konsol Manajemen AWS dengan satu tindakan, terlepas dari berapa banyak contoh yang ada di cluster.

Saat klaster DB dihentikan, Anda hanya dikenakan biaya untuk penyimpanan klaster, snapshot manual, dan penyimpanan cadangan otomatis dalam periode penyimpanan yang ditentukan. Anda tidak dikenakan biaya untuk jam instans DB.

Setelah tujuh hari, Neptune secara otomatis memulai klaster DB Anda untuk memastikan tidak tertinggal pembaruan pemeliharaan yang diperlukan.

Untuk meminimalkan biaya klaster Neptune yang bermuatan ringan, Anda dapat menghentikan klaster tersebut alih-alih menghapus semua replika baca. Untuk cluster dengan lebih dari satu atau dua instance, sering menghapus dan membuat ulang instans DB hanya praktis menggunakan API atau AWS CLI Neptunus, dan penghapusan juga bisa sulit dilakukan dalam urutan yang benar. Misalnya, Anda harus menghapus semua replika baca sebelum menghapus instans utama untuk menghindari pengaktifan mekanisme failover.

Jangan gunakan memulai dan menghentikan jika Anda perlu menjaga klaster DB tetap berjalan tetapi Anda ingin mengurangi kapasitas. Jika klaster Anda terlalu mahal atau tidak terlalu sibuk, Anda bisa

menghapus satu atau lebih instans DB atau ubah instans DB Anda untuk meggunakan kelas instans lebih kecil, tetapi Anda tidak dapat menghentikan instans DB individu.

Menghentikan cluster DB Neptunus

Ketika Anda tidak akan menggunakannya untuk sementara waktu, Anda dapat menghentikan cluster DB Neptunus yang sedang berjalan, dan kemudian memulainya lagi saat Anda membutuhkannya. Sementara klaster Anda dihentikan, Anda dikenakan biaya untuk penyimpanan klaster, snapshot manual, dan penyimpanan cadangan otomatis dalam jendela penyimpanan yang ditentukan, tapi tidak untuk jam instans DB.

Operasi stop menghentikan semua instans replika baca kluster sebelum menghentikan instans utama, untuk menghindari pengaktifan mekanisme failover.

Menghentikan cluster DB menggunakan Konsol Manajemen AWS

Untuk menggunakan Konsol Manajemen AWS untuk menghentikan cluster Neptunus

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Basis data, lalu pilih salah satu klaster. Anda dapat melakukan operasi penghentian dari halaman ini, atau membuka halaman detail untuk klaster DB yang ingin Anda hentikan.
3. Di Tindakan, pilih Berhenti.

Menghentikan cluster DB menggunakan AWS CLI

Untuk menghentikan instance DB dengan menggunakan AWS CLI, panggil stop-db-clusterperintah, menggunakan `--db-cluster-identifier` parameter untuk mengidentifikasi cluster DB yang ingin Anda hentikan.

Example

```
aws neptune stop-db-cluster --db-cluster-identifier mydbcluster
```

Menghentikan cluster DB menggunakan API manajemen Neptunus

Untuk menghentikan instans DB dengan menggunakan API manajemen Neptunus, panggil DBCluster Stop API dan gunakan `DBClusterIdentifier` parameter untuk mengidentifikasi cluster DB yang ingin Anda hentikan.

Apa yang bisa terjadi saat cluster DB dihentikan

- Anda dapat memulihkannya dari snapshot (lihat Melakukan pemulihan dari Snapshot Klaster DB).
- Anda tidak dapat memodifikasi konfigurasi klaster DB, atau salah satu dari instans DB-nya.
- Anda tidak dapat menambah atau menghapus instans DB dari klaster.
- Anda tidak dapat menghapus klaster jika masih memiliki segala yang terkait dengan instans DB.
- Secara umum, Anda harus memulai kembali klaster DB yang berhenti untuk melakukan sebagian besar tindakan administratif.
- Neptune menerapkan pemeliharaan terjadwal pada klaster Anda yang dihentikan segera setelah dimulai lagi. Ingat setelah tujuh hari, Neptune secara otomatis memulai kembali klaster yang telah dihentikan sehingga tidak jauh tertinggal dalam status pemeliharaan.
- Neptune tidak melakukan pencadangan otomatis kluster DB, karena data yang mendasari tidak berubah saat klaster dihentikan.
- Neptune tidak memperpanjang periode penyimpanan cadangan untuk klaster DB saat klaster dihentikan.

Memulai cluster DB Neptunus yang berhenti

Anda hanya dapat memulai sebuah klaster DB Neptune yang berada dalam keadaan berhenti. Saat Anda memulai klaster tersebut, semua instans DB menjadi tersedia lagi. Klaster menjaga pengaturan konfigurasinya, seperti titik akhir, grup parameter, dan grup keamanan VPC.

Memulai cluster DB yang berhenti menggunakan Konsol Manajemen AWS

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Basis data, lalu pilih salah satu klaster. Anda dapat melakukan operasi mulai dari halaman ini, atau navigasi ke halaman detail untuk klaster DB tersebut mulai dari sini.
3. Di Tindakan, pilih Mulai.

Memulai cluster DB yang berhenti menggunakan AWS CLI

Untuk memulai cluster DB yang dihentikan menggunakan AWS CLI, panggil start-db-clusterperintah menggunakan `--db-cluster-identifier` parameter untuk menentukan cluster DB berhenti yang ingin Anda mulai. Menyediakan nama klaster yang ingin Anda pilih saat membuat klaster DB atau gunakan nama instans DB yang Anda pilih dengan `-cluster` yang ditambahkan ke bagian akhir.

Example

```
aws neptune start-db-cluster --db-cluster-identifier mydbcluster
```

Memulai cluster DB yang dihentikan menggunakan API manajemen Neptunus

Untuk memulai cluster DB Neptunus dengan menggunakan API manajemen Neptunus, panggil DBCluster Start API menggunakan parameter untuk menentukan cluster DB berhenti `DBCluster` yang ingin Anda mulai. Menyediakan nama klaster yang Anda pilih saat membuat klaster DB atau gunakan nama instans DB yang Anda pilih, dengan `-cluster` yang ditambahkan ke bagian akhir.

Kosongkan klaster DB Amazon Neptune menggunakan API reset cepat

API REST reset cepat Neptune memungkinkan Anda mereset grafik Neptune dengan cepat dan mudah, menghapus semua datanya.

Anda dapat melakukan ini dalam notebook Neptunus menggunakan sihir baris %db_reset.

- Dalam kebanyakan kasus, operasi reset cepat selesai dalam beberapa menit. Durasi dapat bervariasi tergantung pada beban pada klaster saat operasi dimulai.
- Operasi reset cepat tidak menghasilkan di I/Os tambahan.
- Ukuran volume penyimpanan tidak menyusut setelah reset cepat. Sebaliknya, penyimpanan yang digunakan kembali sebagai data baru dimasukkan. Ini berarti bahwa ukuran volume snapshot yang diambil sebelum dan sesudah operasi reset cepat akan sama. Ukuran volume klaster yang dipulihkan menggunakan snapshot yang dibuat sebelum dan sesudah operasi reset cepat juga akan sama
- Sebagai bagian dari operasi reset, semua instance di cluster DB dimulai ulang.

> **Note**
>
> Dalam kondisi yang jarang terjadi, restart server ini juga dapat mengakibatkan failover cluster.

> **Important**
>
> Menggunakan reset cepat dapat merusak integrasi klaster DB Neptune Anda dengan layanan lainnya. Contoh:
>
> - Pengaturan ulang cepat menghapus semua data aliran dari basis data Anda dan benar-benar me-reset pengaliran. Ini berarti bahwa konsumen pengaliran Anda mungkin tidak lagi bekerja tanpa konfigurasi baru.
> - Reset cepat menghapus semua metadata tentang sumber daya SageMaker AI yang digunakan oleh Neptunus ML, termasuk pekerjaan dan titik akhir. Mereka terus ada di SageMaker AI, dan Anda dapat terus menggunakan titik akhir SageMaker AI yang ada untuk kueri inferensi Neptunus ML, tetapi manajemen Neptunus Neptunus tidak lagi bekerja dengannya. APIs

- Integrasi seperti full-text-search integrasi dengan ElasticSearch juga dihapus dengan reset cepat, dan harus dibuat kembali secara manual sebelum dapat digunakan lagi.

Untuk menghapus semua data dari klaster DB Neptune menggunakan API

1. Pertama, Anda menghasilkan token yang kemudian dapat Anda gunakan untuk melakukan reset basis data. Langkah ini dimaksudkan untuk membantu mencegah siapa pun dari ketidaksengajaan mereset basis data.

   Anda melakukan ini dengan mengirimkan permintaan `HTTP POST` ke titik akhir `/system` pada instans penulis klaster DB Anda untuk menentukan tindakan `initiateDatabaseReset`.

   Perintah `curl` menggunakan tipe konten JSON akan menjadi:

   ```
   curl -X POST \
     -H 'Content-Type: application/json' \
         https://your_writer_instance_endpoint:8182/system \
     -d '{ "action" : "initiateDatabaseReset" }'
   ```

   Atau, menggunakan tipe konten `x-www-form-urlencoded`:

   ```
   curl -X POST \
     -H 'Content-Type: application/x-www-form-urlencoded' \
         https://your_writer_instance_endpoint:8182/system \
     -d 'action=initiateDatabaseReset '
   ```

   Permintaan `initiateDatabaseReset` mengembalikan token reset dalam respon JSON-nya, seperti ini:

   ```
   {
     "status" : "200 OK",
     "payload" : {
       "token" : "new_token_guid"
     }
   }
   ```

   Token tetap valid selama satu jam (60 menit) setelah dikeluarkan.

Jika anda mengirim permintaan ke instans pembaca atau ke titik akhir status, Neptune akan membuang `ReadOnlyViolationException`.

Jika Anda mengirim beberapa permintaan `initiateDatabaseReset`, hanya token terbaru yang dihasilkan akan valid untuk langkah kedua, di mana Anda benar-benar melakukan reset.

Jika server dimulai ulang tepat setelah permintaan `initiateDatabaseReset`, token yang dihasilkan menjadi tidak valid, dan Anda perlu mengirim permintaan baru untuk mendapatkan token baru.

2. Selanjutnya, Anda mengirim premintaan `performDatabaseReset` dengan token yang Anda dapatkan kembali dari `initiateDatabaseReset` ke titik akhir `/system` pada instans penulis klaster DB Anda. Ini akan menghapus semua data dari klaster DB Anda.

   Perintah `curl` menggunakan tipe konten JSON adalah:

   ```
   curl -X POST \
     -H 'Content-Type: application/json' \
         https://your_writer_instance_endpoint:8182/system \
     -d '{
           "action" : "performDatabaseReset",
           "token" : "token_guid"
         }'
   ```

   Atau, menggunakan tipe konten `x-www-form-urlencoded`:

   ```
   curl -X POST \
     -H 'Content-Type: application/x-www-form-urlencoded' \
         https://your_writer_instance_endpoint:8182/system \
     -d 'action=performDatabaseReset&token=token_guid'
   ```

   Permintaan mengembalikan respon JSON. Jika permintaan diterima, tanggapannya adalah:

   ```
   {
     "status" : "200 OK"
   }
   ```

   Jika token yang Anda kirim tidak cocok dengan token yang dikeluarkan, responnya terlihat seperti ini:

```
{
  "code" : "InvalidParameterException",
  "requestId":"token_guid",
  "detailedMessage" : "System command parameter 'token' : 'token_guid' does not
 match database reset token"
}
```

Jika permintaan diterima dan reset dimulai, server memulai ulang dan menghapus data. Anda tidak dapat mengirim permintaan lain untuk klaster DB sementara sedang direset.

Menggunakan API reset cepat dengan IAM-auth

Jika Anda memiliki IAM-auth yang diaktifkan pada klaster DB Anda, Anda dapat menggunakan awscurl untuk mengirim perintah reset cepat yang diotentikasi menggunakan IAM-auth:

Menggunakan awscurl untuk mengirim permintaan cepat-reset dengan IAM-auth

1. Mengatur variabel lingkungan `AWS_ACCESS_KEY_ID` dan `AWS_SECRET_ACCESS_KEY` dengan benar (dan juga `AWS_SECURITY_TOKEN` jika Anda menggunakan kredensial sementara).
2. Permintaan `initiateDatabaseReset` seperti ini:

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "initiateDatabaseReset" }'
```

3. Permintaan `performDatabaseReset` seperti ini:

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "performDatabaseReset" }'
```

Menggunakan meja kerja Neptune line magic **`%db_reset`** untuk me-reset klaster DB

Meja kerja Neptune mendukung line magic `%db_reset` yang memungkinkan Anda melakukan reset basis data cepat di notebook Neptune.

Jika Anda memanggil magic tanpa parameter apapun, Anda melihat layar menanyakan apakah Anda ingin menghapus semua data dalam klaster Anda, dengan kotak centang meminta Anda untuk mengetahui bahwa data cluster tidak akan lagi tersedia setelah Anda menghapusnya. Pada saat itu, Anda dapat memilih untuk melanjutkan dan menghapus data, atau membatalkan operasi.

Pilihan yang lebih berbahaya adalah mengaktifkan `%db_reset` dengan opsi `--yes` atau `-y`, yang menyebabkan penghapusan akan dilakukan tanpa pemberitahuan lebih lanjut.

Anda juga dapat melakukan reset dalam dua langkah, sama seperti dengan REST API:

```
%db_reset --generate-token
```

Tanggapannya adalah:

```
{
  "status" : "200 OK",
  "payload" : {
    "token" : "new_token_guid"
  }
}
```

Lalu lakukan:

```
%db_reset --token new_token_guid
```

Tanggapannya adalah:

```
{
  "status" : "200 OK"
}
```

Kode kesalahan umum untuk operasi reset cepat

Kode kesalahan Neptune	Status HTTP	Pesan	Contoh
`InvalidParameterException`	400	Parameter perintah sistem '*action*'	Parameter tidak valid

Kode kesalahan Neptune	Status HTTP	Pesan	Contoh
		memiliki nilai yang tidak didukung " *XXX*	
`InvalidParameterException`	400	Terlalu banyak nilai yang disediakan untuk: *action*	Permintaan reset cepat dengan lebih dari satu tindakan dikirim dengan header x-www-form-urlencoded 'Content-type:application/ '
`InvalidParameterException`	400	Duplikasi bidang 'tindakan'	Permintaan reset cepat dengan lebih dari satu tindakan yang dikirim dengan header 'Content-Type: aplikasi/json'
`MethodNotAllowedException`	400	Rute buruk:/*bad_endpoint*	Permintaan dikirim ke titik akhir yang salah
`MissingParameterException`	400	Parameter yang dibutuhkan hilang: [action]	Permintaan reset cepat tidak berisi parameter 'tindakan' yang diperlukan
`ReadOnlyViolationException`	400	Menulis tidak diizinkan pada instans replika baca	Permintaan reset cepat dikirim ke pembaca atau titik akhir status

Kode kesalahan Neptune	Status HTTP	Pesan	Contoh
`AccessDeniedException`	403	Token Autentikasi Hilang	Permintaan reset cepat dikirim tanpa tanda tangan yang benar ke titik akhir DB dengan IAM-auth yang diaktifkan
`ServerShutdownException`	500	Reset basis data sedang berlangsung. Silakan coba lagi kueri setelah klaster tersedia.	Kueri Gremlin/Sparql yang ada dan akan datang gagal.

Menambahkan instance pembaca Neptunus ke Cluster DB

Dalam cluster DB Neptunus, ada satu instans DB utama dan hingga 15 instance pembaca Neptunus. Instans DB utama mendukung operasi baca dan tulis, dan melakukan semua modifikasi data pada volume klaster. Instance pembaca Neptunus terhubung ke volume penyimpanan yang sama dengan instans DB utama dan hanya mendukung operasi baca.

Gunakan instance pembaca untuk menurunkan beban kerja baca dari instans DB utama.

Kami menyarankan Anda mendistribusikan instans utama dan pembaca Neptunus di cluster DB Anda melalui beberapa Availability Zone untuk meningkatkan ketersediaan cluster DB Anda.

Bagian berikut menjelaskan cara membuat instance pembaca di cluster DB Anda.

Membuat instance pembaca Neptunus menggunakan konsol

Setelah membuat instance utama untuk cluster DB Neptunus Anda, Anda dapat menambahkan instance pembaca Neptunus tambahan menggunakan konsol Neptunus.

Untuk membuat instance pembaca Neptunus menggunakan Konsol Manajemen AWS

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Basis data.
3. Pilih cluster DB tempat Anda ingin membuat instance pembaca.
4. Pilih Tindakan, lalu pilih Tambah pembaca.
5. Pada halaman Membuat replika DB instans, tentukan opsi untuk replika Neptune Anda. Tabel berikut menunjukkan pengaturan untuk replica baca Neptune.

Untuk opsi ini…	Lakukan ini
Kelas instans DB	Pilih kelas instans DB yang menentukan persyaratan pemrosesan dan memori untuk replica Neptune. Untuk pencantuman kelas instans DB saat ini yang ditawarkan Neptune di wilayah yang berbeda, lihat halaman pricing Neptune.
Zona ketersediaan	Tentukan Availability Zone. Pilih zona yang berbeda dari instans DB utama. Daftar ini hanya mencakup Availability Zone yang dipetakan oleh grup subnet DB yang Anda untuk klaster DB.
Enkripsi	Mengaktifkan atau menonaktifkan enkripsi.
Sumber replika baca	Pilih pengidentifikasi instans utama untuk membuat replica Neptune.
Pengidentifikasi instans DB	Ketik nama instans yang unik untuk akun Anda di Wilayah yang Anda pilih. Anda dapat memilih untuk menambahkan beberapa kecerdasan ke nama, seperti

Untuk opsi ini…	Lakukan ini
	termasuk Availability Zone yang dipilih, misalnya `neptune-us-east-1c` .
Port basis data	Nomor port tempat database menerima koneksi.
Grup parameter DB	Kelompok parameter untuk instans ini.
Ekspor log	Pilih log yang ingin Anda publikasikan, jika ada.
Upgrade Versi Kecil Otomatis	Pilih Ya jika Anda ingin mengaktifkan replika Neptune untuk menerima pemutakhiran versi mesin DB Neptune minor secara otomatis apabila tersedia. Opsi Pemutakhiran Versi Minor Otomatis berlaku hanya pemutakhiran minor. Ini tidak berlaku untuk patch perawatan mesin, yang selalu diterapkan secara otomatis untuk menjaga stabilitas sistem.

6. Pilih Membuat replika baca untuk membuat instans replika Neptune.

Untuk menghapus instance pembaca Neptunus dari kluster DB, ikuti petunjuk di Menghapus instans DB di Amazon Neptunus.

Memodifikasi Klaster DB Neptune Menggunakan Konsol

Ketika Anda memodifikasi instans DB menggunakan Konsol Manajemen AWS, Anda dapat memilih untuk menerapkan perubahan segera dengan memilih Terapkan Segera. Jika Anda memilih untuk menerapkan perubahan langsung, perubahan baru dan perubahan apa pun di antrean modifikasi yang tertunda akan diterapkan sekaligus.

Jika Anda tidak memilih untuk menerapkan perubahan dengan serta-merta, perubahan akan dimasukkan ke dalam antrean perubahan yang tertunda. Selama jendela pemeliharaan berikutnya, perubahan yang tertunda di antrean akan diterapkan.

Important

Jika salah satu dari modifikasi yang tertunda memerlukan waktu henti, memilih terapkan langsung dapat menyebabkan waktu henti yang tidak terduga untuk instans DB di pertanyaan. Tidak ada waktu henti untuk instans DB lain dalam klaster DB.

Note

Ketika Anda mengubah klaster DB di Neptune, pengaturan Terapkan Langsung hanya mempengaruhi Pengidentifikasi klaster DB,Autentikasi IAM DB. Semua modifikasi lainnya akan diterapkan dengan segera, tanpa mempertimbangkan nilai pengaturan Terapkan langsung.

Untuk mengubah klaster DB menggunakan konsol

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Klaster, lalu pilih klaster DB yang ingin Anda modifikasi.
3. Pilih Tindakan, dan kemudian pilih Ubah klaster. Halaman Modifikasi klaster DB akan muncul.
4. Ubah pengaturan apa pun yang Anda inginkan.

   Note

   Di konsol, beberapa perubahan tingkat instans hanya berlaku untuk instans DB saat ini, sedangkan perubahan lainnya berlaku untuk seluruh klaster DB. Untuk mengubah

pengaturan yang memodifikasi seluruh klaster DB pada tingkat instans di konsol, ikuti petunjuk di Memodifikasi instans DB dalam Klaster DB.

5. Jika semua perubahan sesuai keinginan Anda, pilih Lanjutkan dan periksa ringkasan.
6. Untuk menerapkan perubahan dengan segera, pilih Terapkan segera.
7. Di halaman konfirmasi, tinjau perubahan Anda. Jika sudah benar, pilih Ubah klaster untuk menyimpan perubahan Anda.

   Untuk mengedit perubahan Anda, pilih Kembali, atau batalkan perubahan Anda pilihBatalkan.

Memodifikasi instans DB dalam Klaster DB

Untuk memodifikasi instans DB dalam klaster DB gunakan konsol

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Instans, lalu pilih instans DB yang ingin Anda modifikasi.
3. Pilih Tindakan instans, dan kemudian pilih Modifikasi. Halaman Modifikasi Instans DB akan muncul.
4. Ubah pengaturan apa pun yang Anda inginkan.

   > **Note**
   >
   > Beberapa pengaturan berlaku untuk keseluruhan klaster DB dan harus diubah pada tingkat klaster. Untuk mengubah pengaturan tersebut, ikuti petunjuk di Memodifikasi Klaster DB Neptune Menggunakan Konsol.
   > Dalam Konsol Manajemen AWS, beberapa perubahan tingkat instance hanya berlaku untuk instans DB saat ini, sedangkan yang lain berlaku untuk seluruh cluster DB.

5. Jika semua perubahan sesuai keinginan Anda, pilih Lanjutkan dan periksa ringkasan.
6. Untuk menerapkan perubahan dengan segera, pilih Terapkan segera.
7. Di halaman konfirmasi, tinjau perubahan Anda. Jika sudah benar, pilih Modifikasi Instans DB untuk menyimpan perubahan Anda.

   Untuk mengedit perubahan Anda, pilih Kembali, atau batalkan perubahan Anda, pilihBatalkan.

Performa dan Penskalaan di Amazon Neptune

Klaster dan instans DB Neptune diskalakan pada tiga tingkat yang berbeda:

- Penskalaan Penyimpanan
- Penskalaan Instans
- Penskalaan Baca

Penskalaan Penyimpanan di Neptune

Penyimpanan Neptune otomatis melakukan penskalaan dengan data dalam volume kluster Anda. Seiring pertumbuhan data Anda, penyimpanan volume cluster Anda bertambah, hingga 128 TiB di semua wilayah yang didukung kecuali China dan GovCloud, di mana terbatas pada 64 TiB.

Ukuran volume klaster Anda diperiksa setiap jam untuk menentukan biaya penyimpanan Anda.

Penyimpanan yang dikonsumsi oleh database Neptunus Anda ditagih dalam kenaikan per GB-bulan dan yang dikonsumsi oleh basis data Neptunus Anda, I/Os consumed are billed in per million request increments. You pay only for the storage and I/Os dan Anda tidak perlu menyediakan terlebih dahulu.

Untuk informasi harga, lihat halaman produk Neptune.

Penskalaan Instans di Neptune

Anda dapat menskalakan klaster Neptune DB sesuai kebutuhan dengan memodifikasi kelas instans DB untuk setiap instans DB dalam klaster DB. Neptune mendukung beberapa kelas instans DB yang dioptimalkan.

Penskalaan Baca di Neptune

Anda dapat mencapai penskalaan baca untuk klaster DB Neptune dengan membuat hingga 15 replika Neptune dalam klaster DB. Setiap replika Neptune menghasilkan data yang sama dari volume klaster dengan sedikit penundaan replika (sering kali kurang dari 100 milidetik setelah instans utama menulis pembaruan). Saat lalu lintas baca meningkat, Anda dapat membuat replika Neptune tambahan dan terhubung langsung ke mereka untuk mendistribusikan beban baca untuk klaster DB Anda. Replika Neptune tidak harus dari kelas instans DB yang sama dengan instans utama.

Untuk informasi tentang menambahkan replika Neptune ke klaster DB, lihat Menambahkan instance pembaca.

Penskalaan otomatis jumlah replika di cluster DB Amazon Neptunus

Anda dapat menggunakan auto-scaling Neptunus untuk secara otomatis menyesuaikan jumlah replika Neptunus dalam cluster DB untuk memenuhi persyaratan konektivitas dan beban kerja Anda. Penskalaan otomatis memungkinkan cluster DB Neptunus Anda menangani peningkatan beban kerja, dan kemudian, ketika beban kerja berkurang, auto-scaling menghapus replika yang tidak perlu sehingga Anda tidak membayar untuk kapasitas yang tidak digunakan.

Anda hanya dapat menggunakan auto-scaling dengan cluster DB Neptunus yang sudah memiliki satu instance penulis utama dan setidaknya satu instance read-replica (lihat). Klaster dan Instans DB Amazon Neptune Selain itu, semua instance read-replica di cluster harus dalam keadaan tersedia. Jika ada replika baca dalam keadaan selain tersedia, penskalaan otomatis Neptunus tidak melakukan apa pun sampai setiap replika baca di cluster tersedia.

Lihat Buat cluster Neptunus apakah Anda perlu membuat cluster baru.

Dengan menggunakan AWS CLI, Anda menentukan dan menerapkan kebijakan penskalaan ke cluster DB. Anda juga dapat menggunakan AWS CLI untuk mengedit atau menghapus kebijakan auto-scaling Anda. Kebijakan menentukan parameter auto-scaling berikut:

- Jumlah minimum dan maksimum replika yang ada di cluster.
- `ScaleOutCooldown`Interval antara aktivitas penskalaan penambahan replika, dan `ScaleInCooldown` interval antara aktivitas penskalaan penghapusan replika.
- CloudWatch Metrik dan nilai pemicu metrik untuk penskalaan naik atau turun.

Frekuensi tindakan auto-scaling Neptunus diredam dalam beberapa cara:

- Awalnya, untuk auto-scaling untuk menambah atau menghapus pembaca, `CPUUtilization` alarm tinggi harus dilanggar setidaknya selama 3 menit atau alarm rendah harus dilanggar setidaknya selama 15 menit.
- Setelah penambahan atau penghapusan pertama itu, frekuensi tindakan auto-scaling Neptunus berikutnya dibatasi oleh pengaturan dan dalam kebijakan penskalaan otomatis. `ScaleOutCooldown` `ScaleInCooldown`

Jika CloudWatch metrik yang Anda gunakan mencapai ambang batas tinggi yang Anda tentukan dalam kebijakan Anda, dan jika `ScaleOutCooldown` interval telah berlalu sejak tindakan auto-

scaling terakhir, dan jika cluster DB Anda belum memiliki jumlah replika maksimum yang Anda tetapkan, Neptunus auto-scaling akan membuat replika baru menggunakan jenis instance yang sama dengan instance utama cluster DB.

Demikian pula, jika metrik mencapai ambang rendah yang Anda tentukan dan jika `ScaleInCooldown` interval telah berlalu sejak tindakan auto-scaling terakhir, dan jika cluster DB Anda memiliki lebih dari jumlah minimum replika yang Anda tentukan, auto-scaling Neptunus menghapus salah satu replika.

> **Note**
>
> Neptunus auto-scaling hanya menghapus replika yang dibuatnya. Itu tidak menghapus replika yang sudah ada sebelumnya.

Dengan menggunakan parameter cluster neptune_autoscaling_config DB, Anda juga dapat menentukan jenis instance replika baca baru yang dibuat oleh auto-scaling Neptunus, jendela pemeliharaan untuk replika baca tersebut, dan tag yang akan dikaitkan dengan masing-masing replika baca baru. Anda memberikan pengaturan konfigurasi ini dalam string JSON sebagai nilai `neptune_autoscaling_config` parameter, seperti ini:

```
"{
  \"tags\": [
    { \"key\" : \"reader tag-0 key\", \"value\" : \"reader tag-0 value\" },
    { \"key\" : \"reader tag-1 key\", \"value\" : \"reader tag-1 value\" },
  ],
  \"maintenanceWindow\" : \"wed:12:03-wed:12:33\",
  \"dbInstanceClass\" : \"db.r5.xlarge\"
}"
```

Perhatikan bahwa tanda kutip dalam string JSON semua harus lolos dengan karakter garis miring terbalik (). \ Semua spasi dalam string adalah opsional, seperti biasa.

Salah satu dari tiga pengaturan konfigurasi yang tidak ditentukan dalam `neptune_autoscaling_config` parameter disalin dari konfigurasi instance penulis utama cluster DB.

Ketika auto-scaling menambahkan instance read-replica baru, itu akan mengawali ID instans DB dengan (misalnya,). `autoscaled-reader` `autoscaled-reader-7r7t7z3lbd-20210828` Ini juga menambahkan tag ke setiap replika baca yang dibuatnya dengan kunci `autoscaled-reader`

dan nilai. TRUE Anda dapat melihat tag ini pada tab Tag pada halaman detail instans DB di Konsol Manajemen AWS.

```
"key" : "autoscaled-reader",  "value" : "TRUE"
```

Tingkat promosi dari semua instance read-replica yang dibuat oleh auto-scaling adalah prioritas terendah, yang secara default. 15 Ini berarti bahwa selama failover, replika apa pun dengan prioritas yang lebih tinggi, seperti yang dibuat secara manual, akan dipromosikan terlebih dahulu. Lihat Toleransi kesalahan untuk cluster DB Neptunus.

Neptune auto-scaling diimplementasikan menggunakan Application Auto Scaling dengan kebijakan penskalaan pelacakan target yang menggunakan metrik Neptunus sebagai metrik yang telah ditentukan sebelumnya. `CPUUtilization` CloudWatch

Menggunakan auto-scaling di cluster DB tanpa server Neptunus

Neptune Serverless merespons jauh lebih cepat daripada auto-scaling Neptunus ketika permintaan melebihi kapasitas instans, dan meningkatkan skala instance alih-alih menambahkan instance lain. Jika auto-scaling dirancang agar sesuai dengan kenaikan atau penurunan beban kerja yang relatif stabil, tanpa server unggul dalam menangani lonjakan cepat dan kegelisahan dalam permintaan.

Memahami kekuatan mereka, Anda dapat menggabungkan auto-scaling dan serverless untuk menciptakan infrastruktur fleksibel yang akan menangani perubahan beban kerja Anda secara efisien dan memenuhi permintaan sambil meminimalkan biaya.

Agar auto-scaling bekerja secara efektif bersama tanpa server, penting untuk mengonfigurasi pengaturan klaster `maxNCU` tanpa server Anda yang cukup tinggi untuk mengakomodasi lonjakan dan perubahan permintaan singkat. Jika tidak, perubahan transien tidak memicu penskalaan tanpa server, yang dapat menyebabkan auto-scaling memutar banyak instance tambahan yang tidak perlu. Jika `maxNCU` disetel cukup tinggi, penskalaan tanpa server dapat menangani perubahan tersebut lebih cepat dan lebih murah.

Cara mengaktifkan auto-scaling untuk Amazon Neptunus

Penskalaan otomatis hanya dapat diaktifkan untuk cluster DB Neptunus menggunakan file. AWS CLI Anda tidak dapat mengaktifkan auto-scaling menggunakan file. Konsol Manajemen AWS

Selain itu, penskalaan otomatis tidak didukung di wilayah Amazon berikut:

- Afrika (Cape Town): `af-south-1`

- Timur Tengah (UEA): `me-central-1`
- AWS GovCloud (AS-Timur): `us-gov-east-1`
- AWS GovCloud (AS-Barat): `us-gov-west-1`

Mengaktifkan auto-scaling untuk cluster DB Neptunus melibatkan tiga langkah:

1. Daftarkan cluster DB Anda dengan Application Auto Scaling

Langkah pertama dalam mengaktifkan auto-scaling untuk cluster Neptune DB adalah mendaftarkan cluster dengan Application Auto Scaling, menggunakan atau salah satu Application Auto Scaling. AWS CLI SDKs Cluster harus sudah memiliki satu instance utama dan setidaknya satu instance read-replica:

Misalnya, untuk mendaftarkan cluster yang akan diskalakan otomatis dengan dari satu hingga delapan replika tambahan, Anda dapat menggunakan AWS CLI `register-scalable-target`perintah sebagai berikut:

```
aws application-autoscaling register-scalable-target \
  --service-namespace neptune \
  --resource-id cluster:(your DB cluster name) \
  --scalable-dimension neptune:cluster:ReadReplicaCount \
  --min-capacity 1 \
  --max-capacity 8
```

Ini setara dengan menggunakan operasi `RegisterScalableTarget`Application Auto Scaling API.

AWS CLI `register-scalable-target`Perintah mengambil parameter berikut:

- **`service-namespace`** – Atur ke `neptune`.

  Parameter ini setara dengan `ServiceNamespace` parameter di Application Auto Scaling API.
- **`resource-id`**— Setel ini ke pengidentifikasi sumber daya untuk cluster DB Neptunus Anda. Jenis sumber daya adalah`cluster`, yang diikuti oleh titik dua (`:`), dan kemudian nama cluster DB Anda.

  Parameter ini setara dengan `ResourceID` parameter di Application Auto Scaling API.
- **`scalable-dimension`**— Dimensi yang dapat diskalakan dalam hal ini adalah jumlah instance replika di cluster DB, jadi Anda mengatur parameter ini ke. `neptune:cluster:ReadReplicaCount`

Parameter ini setara dengan `ScalableDimension` parameter di Application Auto Scaling API.

- **`min-capacity`**— Jumlah minimum instance replika DB pembaca yang akan dikelola oleh Application Auto Scaling. Nilai ini harus diatur dalam kisaran dari 0 hingga 15, dan harus sama dengan atau kurang dari nilai yang ditentukan untuk jumlah maksimum Replika Neptunus di. `max-capacity` Harus ada setidaknya satu pembaca di cluster DB agar auto-scaling berfungsi.

  Parameter ini setara dengan `MinCapacity` parameter di Application Auto Scaling API.

- **`max-capacity`**— Jumlah maksimum instans replika DB pembaca di cluster DB, termasuk instans yang sudah ada sebelumnya dan instans baru yang dikelola oleh Application Auto Scaling. Nilai ini harus diatur dalam kisaran dari 0 hingga 15, dan harus sama dengan atau lebih besar dari nilai yang ditentukan untuk jumlah minimum Replika Neptunus di. `min-capacity`

  `max-capacity` AWS CLI Parameter ini setara dengan `MaxCapacity` parameter di Application Auto Scaling API.

Saat Anda mendaftarkan cluster DB Anda, Application Auto Scaling membuat peran terkait `AWSServiceRoleForApplicationAutoScaling_NeptuneCluster` layanan. Untuk informasi selengkapnya, lihat Peran terkait layanan untuk auto-scaling Aplikasi di Panduan Pengguna Application Auto Scaling.

2. Tentukan kebijakan penskalaan otomatis untuk digunakan dengan cluster DB Anda

Kebijakan penskalaan pelacakan target didefinisikan sebagai objek teks JSON yang juga dapat disimpan dalam file teks. Untuk Neptunus kebijakan ini saat ini hanya dapat menggunakan metrik `CPUUtilization` CloudWatch Neptunus sebagai metrik yang telah ditentukan sebelumnya. `NeptuneReaderAverageCPUUtilization`

Berikut adalah contoh kebijakan konfigurasi penskalaan pelacakan target untuk Neptunus:

```
{
  "PredefinedMetricSpecification": { "PredefinedMetricType":
 "NeptuneReaderAverageCPUUtilization" },
  "TargetValue": 60.0,
  "ScaleOutCooldown" : 600,
  "ScaleInCooldown" : 600
}
```

`TargetValue`Elemen di sini berisi persentase pemanfaatan CPU di atas yang skala auto-scaling keluar (yaitu, menambahkan lebih banyak replika) dan di bawahnya diskalakan (yaitu, menghapus replika). Dalam hal ini, persentase target yang memicu penskalaan adalah%. `60.0`

`ScaleInCooldown`Elemen menentukan jumlah waktu, dalam detik, setelah aktivitas scale-in selesai sebelum scale-in lain dapat dimulai. Waktu default adalah 300 detik. Di sini, nilai 600 menentukan bahwa setidaknya sepuluh menit harus berlalu antara penyelesaian satu penghapusan replika dan awal yang lain.

`ScaleOutCooldown`Elemen menentukan jumlah waktu, dalam detik, setelah aktivitas scale-out selesai sebelum scale-out lainnya dapat dimulai. Waktu default adalah 300 detik. Di sini, nilai 600 menentukan bahwa setidaknya sepuluh menit harus berlalu antara penyelesaian satu penambahan replika dan awal yang lain.

`DisableScaleIn`Elemennya adalah Boolean yang jika ada dan disetel untuk `true` menonaktifkan scale-in sepenuhnya, yang berarti bahwa auto-scaling dapat menambahkan replika tetapi tidak akan pernah menghapusnya. Secara default, scale-in diaktifkan, dan `DisableScaleIn` sedang. `false`

Setelah mendaftarkan cluster DB Neptune Anda dengan Application Auto Scaling dan menentukan kebijakan penskalaan JSON dalam file teks, selanjutnya terapkan kebijakan penskalaan ke cluster DB terdaftar. Anda dapat menggunakan AWS CLI `put-scaling-policy`perintah untuk melakukan ini, dengan parameter seperti berikut:

```
aws application-autoscaling put-scaling-policy \
  --policy-name (name of the scaling policy) \
  --policy-type TargetTrackingScaling \
  --resource-id cluster:(name of your Neptune DB cluster) \
  --service-namespace neptune \
  --scalable-dimension neptune:cluster:ReadReplicaCount \
  --target-tracking-scaling-policy-configuration file://(path to the JSON configuration
 file)
```

Bila Anda telah menerapkan kebijakan auto-scaling, auto-scaling diaktifkan pada cluster DB Anda.

Anda juga dapat menggunakan AWS CLI `put-scaling-policy`perintah untuk memperbarui kebijakan auto-scaling yang ada.

Lihat juga PutScalingPolicydi Referensi API Application Auto Scaling.

Menghapus auto-scaling dari cluster DB Neptunus

Untuk menghapus auto-scaling dari cluster DB Neptunus, gunakan perintah and. AWS CLI delete-scaling-policyderegister-scalable-target

Mempertahankan Cluster DB Amazon Neptunus Anda

Neptunus melakukan pemeliharaan secara berkala pada semua sumber daya yang digunakannya, termasuk:

- Mengganti perangkat keras yang mendasarinya seperlunya. Ini terjadi di latar belakang, tanpa Anda harus mengambil tindakan apa pun, dan umumnya tidak berpengaruh pada operasi Anda.
- Memperbarui sistem operasi yang mendasarinya. Upgrade sistem operasi dari instans di cluster DB Anda dilakukan untuk meningkatkan kinerja dan keamanan, jadi Anda biasanya harus menyelesaikannya sesegera mungkin. Pembaruan biasanya memerlukan waktu sekitar 10 menit. Pembaruan sistem operasi tidak akan mengubah versi mesin DB atau kelas instans DB dari instans DB.

  Biasanya yang terbaik adalah memperbarui instance pembaca di cluster DB terlebih dahulu, dan kemudian instance penulis. Memperbarui pembaca dan penulis pada saat yang sama dapat menyebabkan downtime jika terjadi failover. Perhatikan bahwa instans DB tidak secara otomatis dicadangkan sebelum pembaruan sistem operasi, jadi pastikan untuk membuat cadangan manual sebelum Anda menerapkan pembaruan sistem operasi.
- Memperbarui mesin database Neptunus. Neptunus secara teratur merilis berbagai pembaruan mesin untuk memperkenalkan fitur dan peningkatan baru dan untuk memperbaiki bug.

Nomor versi mesin

Penomoran versi sebelum rilis mesin 1.3.0.0

Sebelum November 2019, Neptune hanya mendukung satu versi mesin pada satu waktu, dan versi mesin semuanya mengambil bentuk angka, `1.0.1.0.200<xxx>`, saat `xxx` adalah nomor patch. Semua versi mesin baru dirilis sebagai tambalan ke versi sebelumnya.

Pada November 2019, Neptunus mulai mendukung beberapa versi, memungkinkan pelanggan mengontrol jalur peningkatan mereka dengan lebih baik. Akibatnya, penomoran rilis mesin berubah.

Dari November 2019 hingga rilis mesin 1.3.0.0, nomor versi mesin memiliki 5 bagian. Ambil nomor versi `1.0.2.0.R2` sebagai contoh:

- Bagian pertama selalu 1.
- Bagian kedua, `0` di`1.0.2.0.R2`), adalah nomor versi utama database.
- Bagian ketiga dan keempat, `2.0` in`1.0.2.0.R2`) keduanya nomor versi minor.

- Bagian kelima (R2in`1.0.2.0.R2`) adalah nomor patch.

Sebagian besar pembaruan adalah pembaruan tambalan, dan perbedaan antara tambalan dan pembaruan versi minor tidak selalu jelas.

Penomoran versi dari rilis mesin 1.3.0.0 aktif

Dimulai dengan rilis mesin 1.3.0.0, Neptunus mengubah cara pembaruan mesin diberi nomor dan dikelola.

Nomor versi mesin sekarang memiliki empat bagian, yang masing-masing sesuai dengan jenis rilis, sebagai berikut:

product-version.*major-version*.*minor-version*.*patch-version*

Perubahan yang tidak melanggar dari jenis yang sebelumnya dirilis sebagai tambalan sekarang dirilis sebagai versi minor yang dapat Anda kelola menggunakan pengaturan `AutoMinorVersionUpgrade`instans.

Ini berarti bahwa jika Anda mau, Anda dapat menerima pemberitahuan setiap kali versi minor baru dirilis, dengan berlangganan `RDS-EVENT-0156`acara (lihat Berlangganan pemberitahuan acara Neptunus).

Rilis patch sekarang dicadangkan untuk perbaikan bertarget yang mendesak, dan diberi nomor menggunakan bagian terakhir dari nomor versi (`*.*.*.1`,`*.*.*.2`, dan sebagainya).

Berbagai jenis rilis mesin di Amazon Neptunus

Empat jenis pelepasan mesin yang sesuai dengan empat bagian nomor versi mesin adalah sebagai berikut:

- Versi produk — Ini hanya berubah jika produk mengalami perubahan mendasar dalam fungsionalitas atau antarmuka. Versi produk Neptunus saat ini adalah 1.
- Versi utama — Versi utama memperkenalkan fitur baru yang penting dan melanggar perubahan, dan umumnya memiliki masa pakai yang berguna setidaknya dua tahun.
- Versi minor - Versi minor dapat berisi fitur baru, perbaikan, dan perbaikan bug tetapi tidak mengandung perubahan yang melanggar. Anda dapat memilih apakah akan menerapkannya secara otomatis atau tidak selama jendela pemeliharaan berikutnya, dan Anda juga dapat memilih untuk diberi tahu setiap kali dirilis.

- Versi patch — Versi patch dirilis hanya untuk mengatasi perbaikan bug yang mendesak atau pembaruan keamanan kritis. Mereka jarang mengandung perubahan yang melanggar, dan mereka secara otomatis diterapkan selama jendela pemeliharaan berikutnya setelah rilis mereka.

Pembaruan versi utama Amazon Neptunus

Pembaruan versi utama umumnya memperkenalkan satu atau lebih fitur baru yang penting dan sering kali berisi perubahan yang melanggar. Biasanya memiliki masa dukungan sekitar dua tahun. Versi utama Neptunus tercantum dalam rilis Engine, bersama dengan tanggal rilis dan perkiraan akhir masa pakainya.

Pembaruan versi utama sepenuhnya opsional sampai versi utama yang Anda gunakan mencapai akhir masa pakainya. Jika Anda memilih untuk meningkatkan ke versi utama baru, Anda harus menginstal versi baru sendiri menggunakan AWS CLI atau konsol Neptunus seperti yang dijelaskan dalam. Peningkatan versi mayor

Namun, jika versi utama yang Anda gunakan mencapai akhir masa pakainya, Anda akan diberi tahu bahwa Anda diminta untuk meningkatkan ke versi utama yang lebih baru. Kemudian, jika Anda tidak meningkatkan dalam masa tenggang setelah pemberitahuan, peningkatan ke versi utama terbaru secara otomatis dijadwalkan terjadi selama jendela pemeliharaan berikutnya. Untuk informasi selengkapnya, lihat Rentang hidup versi mesin.

Pembaruan versi minor Amazon Neptunus

Sebagian besar pembaruan mesin Neptunus adalah pembaruan versi minor. Mereka terjadi cukup sering dan tidak mengandung perubahan yang melanggar.

Jika Anda memiliki `AutoMinorVersionUpgrade`bidang yang disetel ke `true` dalam instance writer (primer) dari cluster DB Anda, pembaruan versi minor diterapkan secara otomatis ke semua instance di cluster DB Anda selama jendela pemeliharaan berikutnya setelah dirilis.

Jika Anda memiliki `AutoMinorVersionUpgrade`bidang yang disetel ke `false` dalam instance penulis cluster DB Anda, mereka diterapkan hanya jika Anda menginstalnya secara eksplisit.

> **Note**
>
> Pembaruan versi minor bersifat mandiri (tidak bergantung pada pembaruan versi minor sebelumnya ke versi utama yang sama), dan kumulatif (berisi semua fitur dan perbaikan yang

diperkenalkan dalam pembaruan versi minor sebelumnya). Ini berarti Anda dapat menginstal pembaruan versi minor yang diberikan apakah Anda telah menginstal yang sebelumnya atau tidak.

Sangat mudah untuk melacak rilis versi minor dengan berlangganan `RDS-EVENT-0156`acara (lihat Berlangganan pemberitahuan acara Neptunus). Anda kemudian akan diberi tahu setiap kali versi minor baru dirilis.

Juga, apakah Anda berlangganan notifikasi atau tidak, Anda selalu dapat memeriksa untuk melihat pembaruan apa yang tertunda.

Pembaruan versi patch Amazon Neptunus

Dalam kasus masalah keamanan atau cacat serius lainnya yang memengaruhi keandalan instans, Neptunus menyebarkan patch wajib. Mereka diterapkan ke semua instance di cluster DB Anda selama jendela pemeliharaan berikutnya tanpa intervensi apa pun dari Anda.

Rilis patch hanya diterapkan ketika risiko tidak menerapkannya lebih besar daripada risiko dan waktu henti apa pun yang terkait dengan penerapannya. Rilis patch jarang terjadi (biasanya setiap beberapa bulan sekali) dan jarang membutuhkan lebih dari sebagian kecil jendela pemeliharaan Anda untuk diterapkan.

Merencanakan masa pakai versi mesin utama Amazon Neptunus

Versi mesin Neptunus hampir selalu mencapai akhir masa pakainya pada akhir seperempat kalender. Pengecualian hanya terjadi ketika masalah keamanan atau ketersediaan penting muncul.

Ketika versi mesin mencapai akhir masa pakainya, Anda akan diminta untuk meningkatkan basis data Neptunus Anda ke versi yang lebih baru.

Secara umum, versi mesin Neptunus terus tersedia sebagai berikut:

- Versi mesin minor: Versi mesin minor tetap tersedia setidaknya selama 6 bulan setelah dirilis.
- Versi mesin utama: Versi mesin utama tetap tersedia setidaknya selama 12 bulan setelah dirilis.

Setidaknya 3 bulan sebelum versi mesin mencapai akhir masa pakainya, AWS akan mengirimkan pemberitahuan email otomatis ke alamat email yang terkait dengan AWS akun Anda dan memposting

pesan yang sama ke Dasbor AWS Kesehatan Anda. Ini akan memberi Anda waktu untuk merencanakan dan bersiap untuk meningkatkan.

Ketika versi mesin mencapai akhir masa pakainya, Anda tidak akan lagi dapat membuat cluster atau instance baru menggunakan versi itu, juga tidak akan dapat membuat instance menggunakan versi itu.

Versi mesin yang benar-benar mencapai akhir masa pakainya akan secara otomatis ditingkatkan selama jendela pemeliharaan. Pesan yang dikirimkan kepada Anda 3 bulan sebelum akhir masa pakai versi mesin akan berisi detail tentang apa yang akan melibatkan pembaruan otomatis ini, termasuk versi yang akan Anda upgrade secara otomatis, dampaknya pada cluster DB Anda, dan tindakan yang kami rekomendasikan.

Important

Anda bertanggung jawab untuk menjaga versi mesin database Anda tetap terkini. AWS mendesak semua pelanggan untuk meningkatkan basis data mereka ke versi mesin terbaru untuk mendapatkan keuntungan dari perlindungan keamanan, privasi, dan ketersediaan terkini. Jika Anda mengoperasikan database Anda pada mesin atau perangkat lunak yang tidak didukung setelah tanggal penghentian (“Legacy Engine”), Anda menghadapi kemungkinan risiko keamanan, privasi, dan operasional yang lebih besar, termasuk peristiwa downtime.

Pengoperasian database Anda pada mesin apa pun tunduk pada Perjanjian yang mengatur penggunaan AWS Layanan oleh Anda. Mesin Legacy umumnya tidak tersedia. AWS tidak lagi memberikan dukungan untuk Legacy Engine, dan AWS dapat membatasi akses atau penggunaan Legacy Engine kapan saja, jika AWS menentukan Legacy Engine menimbulkan risiko keamanan atau kewajiban, atau risiko bahaya, terhadap Layanan, Afiliasinya AWS, atau pihak ketiga mana pun. Keputusan Anda untuk terus menjalankan Konten Anda di Legacy Engine dapat mengakibatkan Konten Anda menjadi tidak tersedia, rusak, atau tidak dapat dipulihkan. Database yang berjalan pada Legacy Engine tunduk pada Pengecualian Perjanjian Tingkat Layanan (SLA).

DATABASE DAN PERANGKAT LUNAK TERKAIT YANG BERJALAN PADA MESIN LAMA MENGANDUNG BUG, KESALAHAN, CACAT, KOMPONEN AND/OR BERBAHAYA. DENGAN DEMIKIAN, DAN TERLEPAS DARI APA PUN YANG BERTENTANGAN DALAM PERJANJIAN ATAU KETENTUAN LAYANAN, AWS MENYEDIAKAN MESIN LAMA “SEBAGAIMANA ADANYA.”

Mengelola pembaruan mesin ke cluster DB Neptunus Anda

> **Note**
>
> Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan restart database pada instans tersebut, sehingga Anda mengalami downtime mulai dari 20 atau 30 detik hingga beberapa menit, setelah itu Anda dapat melanjutkan menggunakan cluster DB. Pada kesempatan yang jarang terjadi, failover multi-AZ mungkin diperlukan untuk pembaruan pemeliharaan pada instans untuk diselesaikan. Untuk upgrade versi utama yang dapat memakan waktu lebih lama untuk diterapkan, Anda dapat menggunakan strategi penyebaran biru-hijau untuk meminimalkan waktu henti.

Menentukan versi mesin mana yang saat ini Anda gunakan

Anda dapat menggunakan AWS CLI `get-engine-status`perintah untuk memeriksa versi rilis mesin mana yang saat ini digunakan cluster DB Anda:

```
aws neptunedata get-engine-status
```

Output JSON mencakup `"dbEngineVersion"` bidang seperti ini:

```
  "dbEngineVersion": "1.3.0.0",
```

Periksa untuk melihat pembaruan apa yang tertunda dan tersedia

Anda dapat memeriksa pembaruan yang tertunda ke cluster DB Anda menggunakan konsol Neptunus. Pilih Database di kolom kiri dan kemudian pilih cluster DB Anda di panel database. Pembaruan yang tertunda tercantum di kolom Pemeliharaan. Jika Anda memilih Actions dan kemudian Maintenance, Anda memiliki tiga pilihan tentang apa yang harus dilakukan:

- Tingkatkan sekarang.
- Upgrade di jendela berikutnya.
- Tunda peningkatan.

Anda dapat membuat daftar pembaruan mesin yang tertunda menggunakan AWS CLI sebagai berikut:

```
aws neptune describe-pending-maintenance-actions \
  --resource-identifier (ARN of your DB cluster)
  --region (your region) \
  --engine neptune
```

Anda juga dapat membuat daftar pembaruan mesin yang tersedia menggunakan AWS CLI sebagai berikut:

```
aws neptune describe-db-engine-versions \
  --region (your region) \
  --engine neptune
```

Daftar rilis mesin yang tersedia hanya mencakup rilis yang memiliki nomor versi lebih tinggi dari yang sekarang dan yang jalur pemutakhiran ditentukan.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda bahkan tanpa perubahan yang merusak.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan menggunakan solusi penerapan Neptunus Biru-Hijau. Dengan begitu Anda dapat menjalankan aplikasi dan kueri pada versi baru tanpa memengaruhi cluster DB produksi Anda.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri,

serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

Jendela Pemeliharaan Neptune

Jendela perawatan mingguan adalah periode 30 menit di mana pembaruan mesin terjadwal dan perubahan sistem lainnya diterapkan. Sebagian besar acara pemeliharaan selesai selama jendela 30 menit, meskipun acara pemeliharaan yang lebih besar terkadang lebih lama untuk diselesaikan.

Setiap cluster DB memiliki jendela pemeliharaan 30 menit mingguan. Jika Anda tidak menentukan waktu yang disukai untuk itu saat Anda membuat cluster DB, Neptunus secara acak memilih satu hari dalam seminggu dan kemudian secara acak menetapkan periode 30 menit di dalamnya dari blok waktu 8 jam yang bervariasi dengan wilayah.

Di sini, misalnya, adalah blok waktu 8 jam untuk jendela pemeliharaan yang digunakan di beberapa AWS wilayah:

Region	Blok Waktu
Wilayah AS Barat (Oregon)	06.00–14.00 UTC
Wilayah US West (N. California)	06.00–14.00 UTC
Wilayah US East (Ohio)	03:00–11:00 UTC
Wilayah Eropa (Irlandia)	22.00–06.00 UTC

Jendela pemeliharaan menentukan kapan operasi yang tertunda dimulai, dan sebagian besar operasi pemeliharaan selesai di dalam jendela, tetapi tugas pemeliharaan yang lebih besar dapat berlanjut di luar waktu akhir jendela.

Memindahkan jendela pemeliharaan cluster DB Anda

Idealnya, jendela pemeliharaan Anda harus jatuh pada saat Anda cluster berada pada penggunaan terendah. Jika itu tidak benar untuk jendela Anda saat ini, Anda dapat memindahkannya ke waktu yang lebih baik, seperti ini:

Untuk mengubah jendela pemeliharaan cluster DB

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih database.
3. Pilih klaster DB yang periode pemeliharaannya ingin diubah.
4. Pilih Ubah.
5. Pilih Tampilkan lebih banyak di bagian bawah halaman Modify cluster.
6. Di bagian jendela Pemeliharaan pilihan, atur hari, waktu, dan durasi jendela pemeliharaan sesuai keinginan Anda.
7. Pilih Berikutnya.

   Di halaman konfirmasi, tinjau perubahan Anda.
8. Untuk menerapkan perubahan ke periode pemeliharaan secara langsung, pilih Terapkan langsung.
9. Pilih Kirim untuk menerapkan perubahan Anda.

   Untuk mengedit perubahan, pilih Sebelumnya, atau membatalkan perubahan, pilih Batalkan.

Menggunakan AutoMinorVersionUpgrade untuk mengontrol pembaruan versi minor otomatis

> **Important**
>
> `AutoMinorVersionUpgrade`hanya efektif untuk peningkatan versi minor di atas rilis mesin 1.3.0.0.

Jika Anda memiliki `AutoMinorVersionUpgrade` bidang yang disetel ke `true` dalam instance writer (primer) dari cluster DB Anda, pembaruan versi minor diterapkan secara otomatis ke semua instance di cluster DB Anda selama jendela pemeliharaan berikutnya setelah dirilis.

Jika Anda memiliki `AutoMinorVersionUpgrade` bidang yang disetel ke `false` dalam instance penulis cluster DB Anda, mereka diterapkan hanya jika Anda menginstalnya secara eksplisit.

> **Note**
>
> Rilis patch (`*.*.*.1*.*.*.2`,, dll.) selalu diinstal secara otomatis selama jendela pemeliharaan Anda berikutnya, terlepas dari bagaimana `AutoMinorVersionUpgrade` parameter diatur.

Anda dapat mengatur `AutoMinorVersionUpgrade` menggunakan Konsol Manajemen AWS sebagai berikut:

Untuk mengatur **`AutoMinorVersionUpgrade`** menggunakan konsol Neptunus

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Basis data.
3. Pilih instance utama (penulis) dari cluster DB yang ingin Anda atur`AutoMinorVersionUpgrade`.
4. Pilih Ubah.
5. Pilih Tampilkan lebih banyak di bagian bawah halaman Modify cluster.
6. Di bagian bawah halaman yang diperluas, pilih Aktifkan peningkatan versi minor otomatis atau Matikan peningkatan versi minor otomatis.
7. Pilih Berikutnya.

   Di halaman konfirmasi, tinjau perubahan Anda.
8. Untuk menerapkan perubahan pada peningkatan versi minor otomatis, pilih Terapkan segera.
9. Pilih Kirim untuk menerapkan perubahan Anda.

   Untuk mengedit perubahan, pilih Sebelumnya, atau membatalkan perubahan, pilih Batalkan.

Anda juga dapat menggunakan AWS CLI untuk mengatur `AutoMinorVersionUpgrade` bidang. Misalnya, untuk mengaturnya`true`, Anda dapat menggunakan perintah seperti ini:

```
aws neptune modify-db-instance \
  --db-instance-identifier (the ID of your cluster's writer instance) \
  --auto-minor-version-upgrade \
  --apply-immediately
```

Demikian pula, untuk mengaturnya`false`, gunakan perintah seperti ini:

```
aws neptune modify-db-instance \
  --db-instance-identifier (the ID of your cluster's writer instance) \
  --no-auto-minor-version-upgrade \
  --apply-immediately
```

Menginstal pembaruan ke mesin Neptunus Anda secara manual

Menginstal upgrade mesin versi utama

Rilis mesin utama harus selalu dipasang secara manual. Untuk meminimalkan waktu henti dan menyediakan banyak waktu untuk pengujian dan validasi, cara terbaik untuk menginstal versi utama baru umumnya dengan menggunakan solusi penyebaran Biru-Hijau Neptunus.

Dalam beberapa kasus Anda juga dapat menggunakan CloudFormation template yang Anda gunakan untuk membuat cluster DB Anda untuk menginstal upgrade versi utama (lihatMenggunakan CloudFormation template untuk memperbarui versi mesin Cluster DB Neptunus Anda).

Jika Anda ingin segera menginstal pembaruan versi utama, Anda dapat menggunakan perintah CLI seperti berikut:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (identifier for your neptune cluster) \
  --engine neptune \
  --engine-version (the new engine version) \
  --apply-immediately
```

Pastikan untuk menentukan versi mesin yang ingin Anda tingkatkan. Jika tidak, mesin Anda dapat ditingkatkan ke versi yang bukan yang terbaru atau yang Anda harapkan.

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menentukan menggunakan paramater ini:

```
  --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menentukannya menggunakan parameter ini:

```
---db-instance-parameter-group-name (name of the custom instance parameter group)
```

Menginstal upgrade mesin versi minor menggunakan Konsol Manajemen AWS

Untuk melakukan upgrade versi minor menggunakan konsol Neptunus

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Databases, lalu pilih cluster DB yang ingin Anda modifikasi.
3. Pilih Ubah.
4. Di bawah Spesifikasi instans, pilih versi baru yang ingin Anda tingkatkan.
5. Pilih Berikutnya.
6. Jika Anda ingin segera menerapkan perubahan, pilih Terapkan segera.
7. Pilih Kirim untuk memperbarui cluster DB Anda.

Menginstal upgrade mesin versi minor menggunakan AWS CLI

Anda dapat menggunakan perintah seperti berikut ini untuk melakukan upgrade versi minor tanpa menunggu jendela pemeliharaan berikutnya:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (your-neptune-cluster) \
  --engine-version (new-engine-version) \
  --apply-immediately
```

Jika Anda memutakhirkan secara manual menggunakan AWS CLI, pastikan untuk menyertakan versi mesin yang ingin Anda tingkatkan. Jika tidak, mesin Anda mungkin ditingkatkan ke versi yang bukan yang terbaru atau yang Anda harapkan.

Upgrade ke engine versi 1.2.0.0 atau lebih tinggi dari versi yang lebih awal dari 1.2.0.0

Engine release 1.2.0.0 memperkenalkan beberapa perubahan signifikan yang dapat membuat upgrade dari versi sebelumnya lebih rumit dari biasanya:

- Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal

dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

- Engine release 1.2.0.0 memperkenalkan format baru untuk membatalkan log. Akibatnya, jika Anda memutakhirkan ke versi 1.2.0.0 atau lebih tinggi dari versi yang lebih awal dari 1.2.0.0, `UndoLogListSize`metrik harus di bawah ambang batas tertentu. Jika tidak, tambalan akan berputar kembali dan gagal. Ambang batas didasarkan pada jenis instance: batas default adalah 40k untuk instance 4xlarge atau lebih besar, dan 10k untuk instance yang lebih kecil dari 4xlarge. Jika `UndoLogListSize` melebihi batas saat Anda mencoba memutakhirkan, proses tambalan akan kembali, pemutakhiran akan dibatalkan, dan acara dengan alasannya akan terlihat di halaman acara cluster. Batasan ini dapat berubah karena alasan operasional tanpa peringatan sebelumnya.

  Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba meningkatkan dapat membantu menurunkan batas `UndoLogListSize` di bawah ambang batas yang berlaku. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

  Jika `UndoLogListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya di bawah batas yang diperlukan.
- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, pengaturan jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Menggunakan CloudFormation template untuk memperbarui versi mesin Cluster DB Neptunus Anda

Anda dapat menggunakan kembali template CloudFormation Neptunus yang Anda gunakan untuk membuat Cluster DB Neptunus Anda untuk memperbarui versi mesinnya.

Peningkatan versi mesin Neptunus bisa kecil atau besar. Menggunakan CloudFormation template dapat membantu dengan upgrade versi utama, yang sering berisi perubahan signifikan. Karena upgrade versi mayor dapat berisi perubahan database yang tidak kompatibel dengan aplikasi yang ada, Anda mungkin juga perlu membuat perubahan pada aplikasi Anda saat melakukan upgrade. Selalu uji sebelum memutakhirkan, dan kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda sebelum memutakhirkan.

Perhatikan bahwa Anda harus melakukan upgrade mesin terpisah untuk setiap versi utama. Anda tidak dapat melewati versi utama dan meningkatkan langsung ke versi utama berikut.

Sebelum 17 Mei 2023, jika Anda menggunakan tumpukan CloudFormation Neptunus untuk meningkatkan versi mesin Anda, itu hanya membuat cluster DB kosong baru di tempat Anda saat ini. Namun, pada 17 Mei 2023, tumpukan CloudFormation Neptunus sekarang mendukung peningkatan mesin di tempat yang menyimpan data Anda yang ada.

> **Note**
>
> Jika Anda menggunakan AWS Cloud Development Kit (AWS CDK), pastikan bahwa AWS CDK versi yang digunakan adalah 2.82.0 atau yang lebih baru. Versi sebelum 2.82.0 tidak mendukung peningkatan mesin Neptunus di tempat.

Untuk upgrade versi mayor, template Anda harus menetapkan properti berikut di`DBCluster`:

- `DBClusterParameterGroup`(Kustom atau Default)
- `DBInstanceParameterGroupName`
- `EngineVersion`

Demikian pula, untuk DBInstances melekat pada DBCluster Anda harus mengatur:

- `DBParameterGroup`(Kustom/Default)

Pastikan bahwa semua grup parameter Anda ditentukan dalam template, apakah mereka default atau kustom.

Dalam kasus grup parameter kustom, pastikan bahwa keluarga grup parameter kustom Anda yang ada kompatibel dengan versi mesin baru. Versi engine lebih awal dari 1.2.0.0 menggunakan keluarga grup parameter`neptune1`, sedangkan rilis engine dari 1.2.0.0 maju memerlukan keluarga grup parameter. `neptune1.2` Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

Untuk upgrade versi mesin utama, tentukan grup parameter dengan keluarga yang sesuai di `DBCluster` `DBInstanceParameterGroupName` lapangan.

Grup parameter default harus ditingkatkan ke grup yang kompatibel dengan versi mesin baru.

Perhatikan bahwa Neptunus secara otomatis me-reboot instans DB setelah upgrade engine.

Topik

- Contoh: Peningkatan mesin kecil dari 1.2.0.1 ke 1.2.0.2
- Contoh: Upgrade versi utama dari 1.1.1.0 ke 1.2.0.2 dengan grup parameter default
- Contoh: Peningkatan versi utama dari 1.1.1.0 ke 1.2.0.2 dengan grup parameter khusus
- Contoh: Upgrade versi utama dari 1.1.1.0 ke 1.2.0.2 dengan campuran grup parameter default dan kustom

Contoh: Peningkatan mesin kecil dari 1.2.0.1 ke 1.2.0.2

Temukan cluster DB yang ingin Anda upgrade, dan template yang Anda gunakan untuk membuatnya. Contoh:

```
Description: Base Template to create Neptune Stack with Engine Version 1.2.0.1 using
 custom Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1.2
```

```
      Description: test-cfn-neptune-db-cluster-parameter-group-description
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-parameter-group-description
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.1
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Perbarui `EngineVersion` properti dari `1.2.0.1` ke`1.2.0.2`:

```
Description: Template to upgrade minor engine version to 1.2.0.2
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
```

```
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-cluster-parameter-group-description
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-parameter-group-description
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Sekarang gunakan CloudFormation untuk menjalankan template yang direvisi.

Contoh: Upgrade versi utama dari 1.1.1.0 ke 1.2.0.2 dengan grup parameter default

Temukan `DBCluster` yang ingin Anda tingkatkan, dan templat yang Anda gunakan untuk membuatnya. Contoh:

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 default Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

- Perbarui default `DBClusterParameterGroup` ke yang ada di keluarga grup parameter yang digunakan oleh versi mesin baru (di sini`default.neptune1.2`).
- Untuk setiap yang `DBInstance` dilampirkan ke`DBCluster`, perbarui default `DBParameterGroup` ke yang ada di keluarga yang digunakan oleh versi mesin baru (di sini`default.neptune1.2`).
- Setel `DBInstanceParameterGroupName` properti ke grup parameter default dalam keluarga itu (di sini`default.neptune1.2`).
- Perbarui `EngineVersion` properti dari `1.1.0.0` ke`1.2.0.2`.

Template akan terlihat seperti ini:

```
Description: Template to upgrade major engine version to 1.2.0.2 by using upgraded
 default parameter groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName: default.neptune1.2
      DBInstanceParameterGroupName: default.neptune1.2
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
```

Sekarang gunakan CloudFormation untuk menjalankan template yang direvisi.

Contoh: Peningkatan versi utama dari 1.1.1.0 ke 1.2.0.2 dengan grup parameter khusus

Temukan `DBCluster` yang ingin Anda tingkatkan, dan templat yang Anda gunakan untuk membuatnya. Contoh:

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 custom Parameter Groups
Parameters:
```

```
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Name: engineupgradetestcpg
      Family: neptune1
      Description: 'NeptuneDBClusterParameterGroup with family neptune1'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Name: engineupgradetestpg
      Family: neptune1
      Description: 'NeptuneDBParameterGroup1 with family neptune1'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
```

```
    Value:
      Ref: NeptuneDBCluster
```

- Perbarui `DBClusterParameterGroup` keluarga khusus ke yang digunakan oleh versi mesin baru di sini`default.neptune1.2`).
- Untuk setiap yang `DBInstance` dilampirkan pada`DBCluster`, perbarui `DBParameterGroup` keluarga khusus ke yang digunakan oleh versi mesin baru (di sini`default.neptune1.2`).
- Setel `DBInstanceParameterGroupName` properti ke grup parameter dalam keluarga itu (di sini`default.neptune1.2`).
- Perbarui `EngineVersion` properti dari `1.1.0.0` ke`1.2.0.2`.

Template akan terlihat seperti ini:

```
Description: Template to upgrade major engine version to 1.2.0.2 by modifying existing
 custom parameter groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Name: engineupgradetestcpgnew
      Family: neptune1.2
      Description: 'NeptuneDBClusterParameterGroup with family neptune1.2'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Name: engineupgradetestpgnew
      Family: neptune1.2
      Description: 'NeptuneDBParameterGroup1 with family neptune1.2'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
```

```
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
      DBInstanceParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Sekarang gunakan CloudFormation untuk menjalankan template yang direvisi.

Contoh: Upgrade versi utama dari 1.1.1.0 ke 1.2.0.2 dengan campuran grup parameter default dan kustom

Temukan `DBCluster` yang ingin Anda tingkatkan, dan templat yang Anda gunakan untuk membuatnya. Contoh:

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 custom Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
```

```
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1
      Description: 'NeptuneDBClusterParameterGroup with family neptune1'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1
      Description: 'NeptuneDBParameterGroup with family neptune1'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  CustomNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
  DefaultNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
```

```
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

- Untuk grup parameter cluster khusus, perbarui `DBClusterParameterGroup` keluarga ke yang sesuai dengan versi mesin baru, yaitu`neptune1.2`.
- Untuk grup parameter cluster default, perbarui `DBClusterParameterGroup` ke default yang sesuai dengan versi mesin baru, yaitu`default.neptune1.2`.
- Untuk setiap yang `DBInstance` dilampirkan ke`DBCluster`, perbarui default `DBParameterGroup` ke yang ada di keluarga yang digunakan oleh versi mesin baru (di sini`default.neptune1.2`), dan grup parameter khusus ke grup yang menggunakan keluarga yang didukung oleh versi mesin baru (di sini`neptune1.2`).
- Atur `DBInstanceParameterGroupName` properti ke grup parameter dalam keluarga yang didukung oleh versi mesin baru.

Template akan terlihat seperti ini:

```
Description: Template to update Neptune Stack to Engine Version 1.2.0.1 using custom
 and default Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1.2
      Description: 'NeptuneDBClusterParameterGroup with family neptune1.2'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: 'NeptuneDBParameterGroup1 with family neptune1.2'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
```

```
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
      DBInstanceParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBClusterParameterGroup
  CustomNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
  DefaultNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Sekarang gunakan CloudFormation untuk menjalankan template yang direvisi.

Kloning Basis Data di Neptune

Menggunakan kloning DB, Anda dapat dengan cepat dan berbiaya-efektif membuat klon dari semua basis data Anda di Amazon Neptune. Basis data klon hanya mensyaratkan ruang tambahan yang minimal saat pertama kali dibuat. Kloning database menggunakan copy-on-write protokol. Data disalin pada saat yang sama dengan diubah, baik di basis data sumber maupun basis data klon. Anda dapat membuat beberapa klon dari kluster DB yang sama. Anda juga dapat membuat klon tambahan dari klon lain. Untuk informasi lebih lanjut tentang cara kerja copy-on-write protokol dalam konteks penyimpanan Neptunus, lihat. Copy-on-Write Protokol

Anda dapat menggunakan kloning DB dalam berbagai kasus penggunaan, khususnya jika Anda tidak ingin memiliki dampak pada lingkungan produksi Anda, seperti berikut ini:

- Lakukan eksperimen dengan dan nilai dampak perubahan, seperti perubahan skema atau perubahan grup parameter.
- Lakukan operasi dengan beban kerja intensif, seperti mengekspor data atau menjalankan kueri analitis.
- Buat salinan klaster DB produksi dalam lingkungan nonproduksi untuk pengembangan atau pengujian.

Untuk membuat klon dari cluster DB menggunakan Konsol Manajemen AWS

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Instans. Pilih instans primer untuk klaster DB yang ingin Anda buat klonnya.
3. Pilih Tindakan Instan, lalu pilih Buat klon.
4. Di halaman Buat Klon, ketik nama untuk instans primer klaster DB klon sebagai Pengidentifikasi instans DB.

   Jika ingin, konfigurasikan pengaturan lain untuk klaster DB klon. Untuk informasi tentang pengaturan klaster DB yang berbeda, lihat Luncurkan menggunakan konsol.
5. Pilih Buat Klon untuk meluncurkan klaster DB klon.

Untuk membuat klon dari cluster DB menggunakan AWS CLI

- Panggil perintah restore-db-cluster-toNeptunus point-in-time AWS CLI - dan berikan nilai-nilai berikut:
  - `--source-db-cluster-identifier` – Nama klaster DB sumber untuk membuat klon.
  - `--db-cluster-identifier` – Nama klaster DB klon.
  - `--restore-type copy-on-write` – Nilai `copy-on-write` menunjukkan bahwa klaster DB klon harus dibuat.
  - `--use-latest-restorable-time` – Ini menentukan bahwa waktu pencadangan terbaru yang dapat dipulihkan harus digunakan.

> **Note**
>
> point-in-time AWS CLI Perintah restore-db-cluster-to- hanya mengkloning cluster DB, bukan instance DB untuk cluster DB itu.

Linux/UNIX Contoh berikut membuat klon dari cluster `source-db-cluster-id` DB dan menamai klon`db-clone-cluster-id`.

```
aws neptune restore-db-cluster-to-point-in-time \
  --region us-east-1 \
  --source-db-cluster-identifier source-db-cluster-id \
  --db-cluster-identifier db-clone-cluster-id \
  --restore-type copy-on-write \
  --use-latest-restorable-time
```

Contoh yang sama bekerja pada Windows jika karakter escape line-end \digantikan oleh yang setara dengan Windows ^:

```
aws neptune restore-db-cluster-to-point-in-time ^
  --region us-east-1 ^
  --source-db-cluster-identifier source-db-cluster-id ^
  --db-cluster-identifier db-clone-cluster-id ^
  --restore-type copy-on-write ^
  --use-latest-restorable-time
```

Batasan

Kloning DB di Neptune memiliki batasan sebagai berikut:

- Anda tidak dapat membuat database klon di seluruh AWS Wilayah. Basis data klon harus dibuat dalam Wilayah yang sama dengan basis data sumber.
- Basis data yang diklonning selalu menggunakan patch terbaru dari versi mesin Neptune yang sedang digunakan oleh basis data yang diklonning. Hal ini benar bahkan jika basis data sumber belum ditingkatkan ke versi patch tersebut. Tetapi, versi mesin itu sendiri tidak berubah.
- Saat ini, Anda dibatasi tidak lebih dari 15 klon per salinan cluster DB Neptunus Anda, termasuk klon berdasarkan klon lain. Setelah mencapai batas itu, Anda harus membuat salinan lain dari database Anda daripada mengkloningnya. Namun, jika Anda membuat salinan baru, itu juga dapat memiliki hingga 15 klon.
- Kloning DB cross-account saat ini tidak didukung.
- Anda dapat menyediakan Virtual Private Cloud (VPC) yang berbeda untuk klon Anda. Namun, subnet di dalamnya VPCs harus dipetakan ke set Availability Zone yang sama.

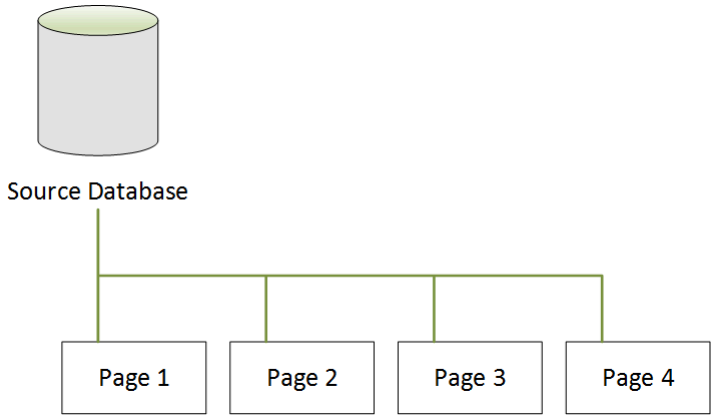
Copy-on-Write Protokol untuk Kloning DB

Skenario berikut menggambarkan bagaimana copy-on-write protokol bekerja.

- Basis Data Neptune Sebelum Kloning
- Basis Data Neptune Setelah Kloning
- Ketika Perubahan Dibuat ke Basis Data Sumber
- Ketika Perubahan Dibuat ke Basis Data Klon

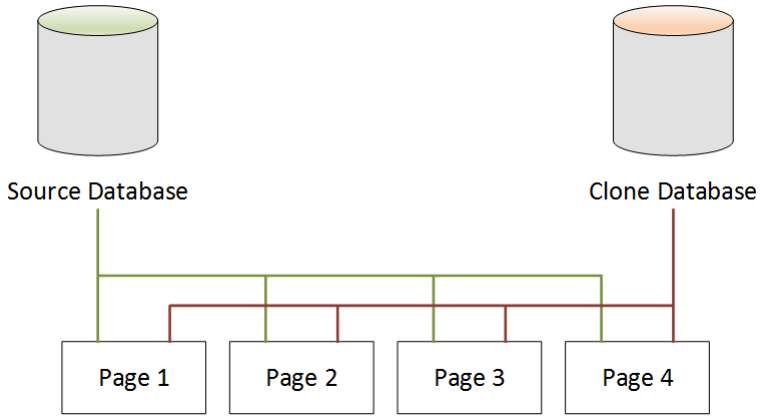
Basis Data Neptune Sebelum Kloning

Data dalam basis data sumber disimpan dalam halaman. Pada diagram berikut, basis data sumber memiliki empat halaman.



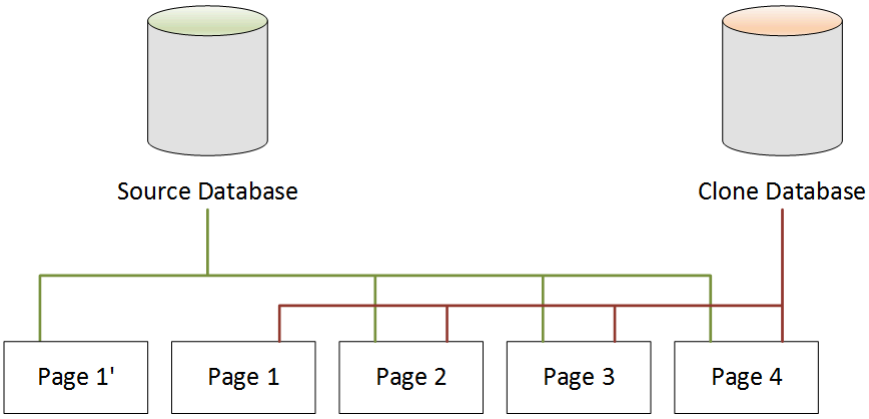
Basis Data Neptune Setelah Kloning

Seperti yang ditunjukkan dalam diagram berikut, tidak ada perubahan pada basis data sumber setelah kloning DB. Baik basis data sumber dan basis data klon menunjuk ke empat halaman yang sama. Tidak ada halaman yang secara fisik disalin, sehingga tidak perlu penyimpanan tambahan.



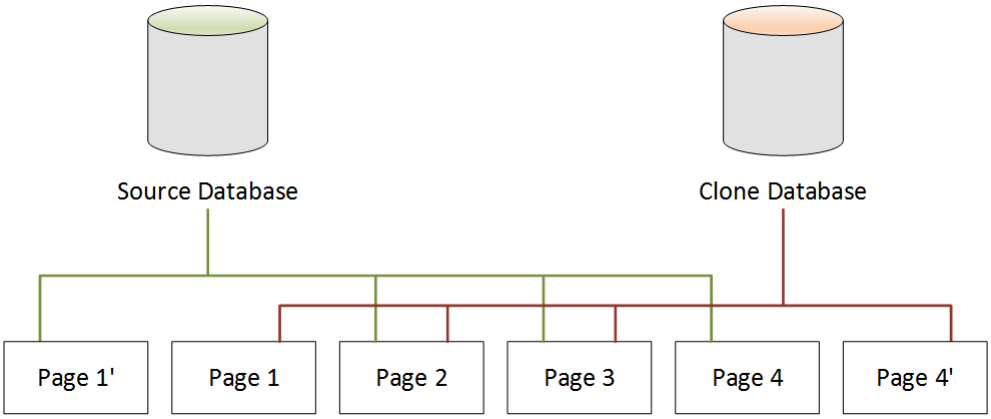
Ketika Perubahan Dibuat ke Basis Data Sumber

Pada contoh berikut, basis data sumber membuat perubahan ke data dalam`Page 1`. Alih-alih menulis ke aslinya `Page 1`, itu menggunakan penyimpanan tambahan untuk membuat halaman baru, disebut `Page 1'`. Basis data sumber sekarang menunjuk ke `Page 1'` baru, dan juga ke `Page 2`, `Page 3`, dan`Page 4`. Basis data klon terus menunjuk ke `Page 1` melalui `Page 4`.



Ketika Perubahan Dibuat ke Basis Data Klon

Pada diagram berikut, basis data klon juga telah berubah, kali ini dalam `Page 4`. Alih-alih menulis ke aslinya `Page 4`, penyimpanan tambahan digunakan untuk membuat halaman baru, disebut `Page 4'`. Basis data sumber terus menunjuk ke `Page 1'`, dan juga `Page 2` melalui `Page 4`, namun basis data klon sekarang menunjuk ke `Page 1`melalui `Page 3`, dan juga `Page 4'`.



Seperti yang ditunjukkan dalam skenario kedua, setelah kloning DB, tidak ada penyimpanan tambahan yang diperlukan pada saat pembuatan klon. Namun, karena perubahan terjadi dalam basis data sumber dan basis data klon, hanya halaman yang diubah yang dibuat, seperti yang ditunjukkan dalam skenario ketiga dan keempat. Karena lebih banyak perubahan yang terjadi seiring waktu baik pada basis data sumber maupun basis data klon, Anda perlu lebih banyak penyimpanan secara bertahap untuk menangkap dan menyimpan perubahan tersebut.

Menghapus Basis Data Sumber

Menghapus basis data sumber tidak mempengaruhi basis data klon database yang terkait dengannya. Basis data klon terus menunjuk ke halaman yang sebelumnya dimiliki oleh basis data sumber.

Mengelola Instans Amazon Neptune

Bagian berikut memiliki informasi tentang operasi tingkat instans.

Topik

- Kelas Instans Burstable Neptune T3.
- Memodifikasi instans DB Neptune (dan Menerapkan Segera)
- Mengganti nama Instans DB Neptune
- Mem-boot ulang instans DB di Amazon Neptunus
- Menghapus sebuah Instans DB di Amazon Neptune

Kelas Instans Burstable Neptune T3.

Sbagai tambahan kelas instans performa tetap seperti R5 dan R6, Amazon Neptune memberi Anda opsi menggunakan performa burstable-instans T3. Sementara Anda sedang mengembangkan aplikasi grafik Anda, Anda ingin basis data Anda menjadi cepat dan responsif, tetapi Anda tidak perlu menggunakannya sepanjang waktu. Kelas instans db.t3.medium Neptune satu-satunya yang harus Anda gunakan dalam situasi itu, dengan biaya secara signifikan lebih rendah daripada kelas instans performa tetap paling mahal.

Instans burstable berjalan pada tingkat dasar performa CPU sampai beban kerja membutuhkan lebih banyak, dan kemudian melakukan burst jauh di atas dasar untuk selama beban kerja memerlukannya. Harga per jam mencakup burstnya, asalkan pemanfaatan CPU rata-rata tidak melebihi baseline selama periode 24 jam. Untuk kebanyakan situasi pengembangan dan pengujian, yaitu menerjemahkan ke performa yang baik dengan biaya rendah.

Jika Anda memulai dengan kelas T3 instance, Anda dapat dengan mudah beralih nanti ke kelas instance performa tetap saat Anda siap untuk masuk ke produksi, menggunakan, Konsol Manajemen AWS AWS CLI, atau salah satu kelas. AWS SDKs

Bursting T3 diatur oleh CPU Credit

Sebuah kredit CPU mewakili pemanfaatan penuh dari satu inti CPU virtual (vCPU) selama satu menit. Itu juga dapat diterjemahkan menjadi 50% pemanfaatan vCPU selama dua menit, atau 25% pemanfaatan dua CPUs v selama dua menit, dan seterusnya.

Instans T3 menghasilkan kredit CPU saat diam dan menggunakannya saat aktif, keduanya diukur pada resolusi milidetik. Kelas db.t3.medium instance memiliki dua vCPUs, yang masing-masing menghasilkan 12 kredit CPU per jam saat idle. Ini berarti bahwa 20% pemanfaatan setiap vCPU menghasilkan saldo kredit CPU nol. Kredit CPU 12 yang diperoleh dihabiskan oleh pemanfaatan 20% vCPU (sejak 20% dari 60 menit juga 12). Dengan demikian, pemanfaatan 20% ini adalah tingkat pemanfaatan dasar yang menghasilkan tidak keseimbangan CPU-credit positif maupun negatif.

Waktu diam (pemanfaatan CPU di bawah 20% dari total yang tersedia) menyebabkan kredit CPU disimpan dalam bucket saldo kredit, sampai batas untuk kelas instans db.t3.medium 576 (jumlah maksimum kredit CPU yang dapat diperoleh dalam 24 jam, yaitu 2 x 12 x 24). Lebih dari batas itu, kredit CPU hanya dibuang.

Bila diperlukan, pemanfaatan CPU dapat meledak hingga setinggi 100% selama dibutuhkan oleh beban kerja, bahkan setelah saldo kredit CPU turun di bawah nol. Jika instans mempertahankan

saldo negatif terus menerus selama 24 jam, itu akan dikenakan biaya tambahan sebesar $0,05 untuk setiap -60 kredit CPU di atas periode tersebut. Namun, untuk sebagian besar pengembangan dan beban kerja pengujian, bursting biasanya dilindungi oleh waktu idle sebelum atau setelah burst.

> **Note**
>
> Kelas T3 instance Neptunus dikonfigurasi seperti mode EC2 tak terbatas Amazon.

Menggunakan Konsol Manajemen AWS untuk Membuat Instance Burstable T3

Dalam Konsol Manajemen AWS, Anda dapat membuat instance cluster DB primer atau instance read-replica yang menggunakan kelas `db.t3.medium` instance, atau Anda dapat memodifikasi instance yang ada untuk menggunakan kelas instance. `db.t3.medium`

Misalnya, untuk membuat instans utama klaster DB baru di konsol Neptune:

- Pilih Buat Basis Data.
- Pilih Versi mesin DB sama dengan atau lebih lambat dari `1.0.2.2`.
- Di bawah Tujuan, pilih Pengembangan dan Pengujian.
- Sebagai Kelas instans DB, terima default: `db.t3.medium — 2 vCPU, 4 GiB RAM`.

Menggunakan AWS CLI untuk Membuat Instance Burstable T3

Anda juga dapat menggunakan AWS CLI untuk melakukan hal yang sama:

```
aws neptune create-db-cluster \
    --db-cluster-identifier (name for a new DB cluster) \
    --engine neptune \
    --engine-version "1.0.2.2"

aws neptune create-db-instance \
    --db-cluster-identifier (name of the new DB cluster) \
    --db-instance-identifier (name for the primary writer instance in the cluster) \
    --engine neptune \
    --db-instance-class db.t3.medium
```

Memodifikasi instans DB Neptune (dan Menerapkan Segera)

Anda dapat menerapkan sebagian besar perubahan ke instans DB Amazon Neptune segera atau menunda hingga waktu pemeliharaan berikutnya. Beberapa modifikasi, seperti perubahan grup parameter, memerlukan boot ulang instans DB secara manual agar dapat diterapkan.

> **Important**
>
> Modifikasi mengakibatkan pemadaman jika Neptunus harus me-reboot instans DB Anda agar perubahan diterapkan. Tinjau dampak pada database dan aplikasi Anda sebelum memodifikasi pengaturan instans DB.

Implikasi Pengaturan Umum dan Waktu Henti

Tabel berikut berisi detail tentang pengaturan mana yang dapat diubah, kapan perubahan dapat diterapkan, dan apakah perubahan menyebabkan waktu henti untuk instans DB.

Pengaturan instans DB	Catatan waktu henti
Kelas instans DB	Pemadaman terjadi selama perubahan ini, apakah itu diterapkan segera atau selama jendela pemeliharaan berikutnya.
Pengidentifikasi instans DB	Instans DB di-boot ulang dan pemadaman terjadi selama perubahan ini, apakah itu diterapkan segera atau selama jendela pemeliharaan berikutnya.
Grup subnet	Instans DB di-boot ulang dan pemadaman terjadi selama perubahan ini, apakah itu diterapkan segera atau

Pengaturan instans DB	Catatan waktu henti	
	selama jendela pemeliharaan berikutnya.	
Grup keamanan	Perubahan diterapkan secara asinkron sesegera mungkin, terlepas dari kapan Anda menentukan perubahan harus dilakukan, dan tidak ada hasil pemadaman.	–
Otoritas Sertifikat	Secara default, instans DB dimulai ulang saat Anda menetapkan Otoritas Sertifika t baru.	
Pelabuhan Database	Perubahan selalu terjadi segera, menyebabkan instans DB di-boot ulang, dan terjadi pemadaman.	

Pengaturan instans DB	Catatan waktu henti	
Grup parameter DB	Mengubah pengaturan ini tidak mengakibatkan pemadaman listrik. Nama grup parameter itu sendiri segera berubah, tetapi perubahan parameter yang sebenarnya tidak diterapka n sampai Anda me-reboot instans tanpa failover. Dalam kasus ini, instans DB tidak di-reboot secara otomatis, dan perubahan parameter tidak diterapkan selama jendela pemeliharaan berikutnya. Namun, jika Anda memodifik asi parameter dinamis dalam grup parameter DB yang baru terkait, perubahan ini diterapkan segera tanpa reboot. Untuk informasi selengkapnya, lihat Mem-boot ulang instans DB di Amazon Neptunus.	
Grup parameter klaster basis data	Nama grup parameter DB segera diubah.	

Pengaturan instans DB	Catatan waktu henti
Periode retensi cadangan	Jika Anda menentukan bahwa perubahan harus segera terjadi, perubahan ini segera terjadi. Jika tidak, jika Anda mengubah pengatura n dari nilai bukan nol ke nilai bukan nol lainnya, perubahan diterapkan secara asinkron, sesegera mungkin. Perubahan lain terjadi selama jendela pemeliharaan berikutnya. Pemadaman terjadi jika Anda mengubah dari nilai 0 ke nilai bukan nol, atau dari nilai bukan nol ke 0.
Log audit	Pilih Log audit jika Anda ingin menggunakan pencatatan audit melalui CloudWatch Log. Anda juga harus mengatur `neptune_enable_aud it_log` parameter dalam grup parameter cluster DB ke `enable` (1) agar pencatatan audit diaktifkan.

Pengaturan instans DB	Catatan waktu henti
Peningkatan versi minor otomatis	Pilih Aktifkan upgrade versi minor otomatis jika Anda ingin mengaktifkan klaster DB Neptune agar menerima upgrade versi mesin minor secara otomatis saat tersedia. Opsi Upgrade versi minor otomatis hanya berlaku untuk upgrade versi mesin minor untuk klaster DB Amazon Neptune Anda. Opsi ini tidak berlaku untuk patch biasa yang diterapkan untuk menjaga stabilitas sistem.

Mengganti nama Instans DB Neptune

Anda dapat mengubah nama instans DB Amazon Neptune dengan menggunakan Konsol Manajemen AWS. Mengganti nama instans DB dapat memiliki efek yang jauh jangkauannya. Berikut ini adalah daftar hal-hal yang harus Anda ketahui sebelum mengubah nama instans DB.

- Saat Anda mengubah nama instance DB, titik akhir untuk instans DB berubah, karena URL mencakup nama yang Anda tetapkan ke instans DB. Anda harus selalu mengalihkan lalu lintas dari URL lama ke yang baru.
- Saat Anda mengganti nama instans DB, nama DNS lama yang digunakan oleh instans DB akan segera dihapus, meskipun tetap disimpan selama beberapa menit. Nama DNS baru untuk instans DB yang diubah namanya menjadi efektif dalam waktu sekitar 10 menit. Instans DB yang diubah namanya tidak tersedia hingga nama baru menjadi efektif.
- Anda tidak dapat menggunakan nama instans DB yang sudah ada saat Anda melakukan penggantian nama suatu instans.
- Semua replika baca terkait dengan suatu instans DB tetap terkait dengan instans tersebut setelah diberi nama ulang. Misalnya, bayangkan Anda memiliki instans DB yang melayani basis data produksi Anda dan instans tersebut memiliki beberapa replika baca terkait. Jika Anda mengganti nama instans DB lalu IT di lingkungan produksi dengan snapshot DB, instans DB yang Anda ubah namanya masih memiliki replika baca yang terkait dengannya.
- Metrik dan peristiwa terkait dengan nama instans DB dipertahankan jika Anda menggunakan ulang nama instans DB. Misalnya, jika Anda mempromosikan replika baca dan mengganti namanya menjadi nama instance utama sebelumnya, peristiwa dan metrik yang dikaitkan dengan instance utama kemudian dikaitkan dengan instance yang diganti namanya.
- Tag instans DB akan dipertahankan dengan instans DB, terlepas dari perubahan namanya.
- Snapshot DB dipertahankan untuk instans DB yang diubah namanya.

Mengganti nama instans DB dengan menggunakan konsol Neptune.

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Basis data.
3. Pilih tombol radio di samping instans DB yang ingin Anda ubah namanya.
4. Di menu Tindakan instans, pilih Memodifikasi.

5. Masukkan nama baru di Pengidentifikasi instans DB kotak teks. Pilih Langsung Terapkan, lalu pilih Lanjutkan.
6. Pilih Modifikasi instans DB untuk menyelesaikan perubahan.

Mem-boot ulang instans DB di Amazon Neptunus

Dalam beberapa kasus, jika Anda memodifikasi instans DB Amazon Neptune, ubah grup parameter DB yang berhubungan dengan instans, atau ubah parameter DB statis dalam grup parameter yang digunakan instans, Anda harus mereboot instans untuk menerapkan perubahan.

Boot ulang instans DB akan mengaktifkan ulang layanan mesin basis data. Reboot juga berlaku untuk instans DB, perubahan apa pun pada grup parameter DB terkait yang tertunda. Mereboot instans DB akan menyebabkan matinya sementara, selama status instans DB diatur ke rebooting. Jika instans Neptune dikonfigurasi untuk Multi-AZ, boot ulang dapat dilakukan melalui failover. Peristiwa Neptune dibuat saat boot ulang selesai.

Jika instans DB Anda deployment Multi-AZ, Anda dapat memaksakan failover dari satu Availability Zone ke yang lain saat Anda memilih opsi Reboot. Saat Anda memaksa failover instans DB Anda, Neptune secara otomatis beralih ke replika siaga di Availability Zone lain. Kemudian memperbarui catatan DNS untuk instans DB untuk menunjuk ke instans DB siaga. Hasilnya, Anda harus membersihkan dan membangun kembali koneksi yang sudah ada ke instans DB Anda.

Reboot dengan failover bermanfaat saat Anda ingin mensimulasikan kegagalan instans DB untuk pengujian, atau memulihkan operasi ke Availability Zone asli setelah failover terjadi. Untuk informasi selengkapnya, lihat Ketersediaan Tinggi (Multi-AZ) di Panduan Pengguna Amazon RDS. Ketika Anda me-reboot klaster DB, klaster akan melakukan failover ke replika siaga. Mereboot replika Neptune tidak memulai failover.

Waktu yang dibutuhkan untuk reboot adalah fungsi dari proses pemulihan crash. Untuk meningkatkan waktu reboot, kami menyarankan agar Anda mengurangi aktivitas basis data sebanyak mungkin selama proses reboot untuk mengurangi aktivitas rollback untuk transaksi in-transit.

Pada konsol, opsi Mulai ulang mungkin dinonaktifkan jika instans DB tidak dalam keadaan Tersedia. Ini bisa karena beberapa alasan, seperti pencadangan yang sedang berlangsung, modifikasi yang diminta konsumen, atau tindakan jendela-pemeliharaan.

> **Note**
>
> SebelumnyaRilis: 1.2.0.0 (2022-07-21), semua replika baca dalam cluster DB secara otomatis di-boot ulang setiap kali instance utama (penulis) dimulai ulang. Dimulai denganRilis: 1.2.0.0 (2022-07-21), memulai ulang instance utama tidak menyebabkan replika apa pun dimulai ulang. Ini berarti bahwa jika Anda mengubah

parameter cluster, Anda harus memulai ulang setiap instance secara terpisah untuk mengambil perubahan parameter (lihatGrup parameter).

Untuk me-reboot instans DB menggunakan konsol Neptune

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Basis data.
3. Pilih instans DB yang ingin Anda reboot.
4. Pilih Tindakan instans, dan kemudian pilih Reboot.
5. Untuk memaksa failover dari satu Availability Zone ke lainnya, pilih Reboot dengan failover? di kotak dialog Reboot instans DB.
6. Pilih Boot ulang. Untuk membatalkan reboot, pilih Batal.

Menghapus sebuah Instans DB di Amazon Neptune

Anda dapat menghapus instans Amazon Neptunus DB di negara bagian mana pun dan kapan saja, selama instans telah dimulai.

> **Warning**
>
> Jika Anda menghapus instance terakhir yang tersisa di cluster menggunakan konsol web, itu juga akan menghapus volume penyimpanan cluster yang mendasarinya.

Mengambil Snapshot Final Instans DB Anda Sebelum Menghapusnya

Untuk menghapus instans DB, tentukan nama instans dan apakah Anda ingin memiliki snapshot DB akhir yang diambil dari instans. Jika instans DB yang Anda hapus memiliki status Membuat, Anda tidak dapat memiliki snapshot DB akhir yang diambil. Jika instans DB berada dalam status kegagalan dengan status gagal, incompatible-restore, atau incompatible-network, Anda hanya dapat menghapus instans ketika parameter `SkipFinalSnapshot` diatur ke `true`.

Jika Anda menghapus semua instans DB Neptunus di cluster DB menggunakan Konsol Manajemen AWS, seluruh cluster DB dihapus secara otomatis. Jika Anda menggunakan AWS CLI atau SDK, Anda harus menghapus cluster DB secara manual setelah Anda menghapus instance terakhir.

> **Important**
>
> Jika Anda menghapus seluruh klaster DB, semua cadangan otomatisnya dihapus pada waktu yang sama dan tidak dapat dipulihkan. Ini berarti kecuali jika Anda memilih untuk membuat snapshot DB akhir secara manual, Anda tidak dapat mengembalikan instans DB ke keadaan akhirnya di lain waktu. Snapshot manual sebiah instans tidak dihapus ketika klaster dihapus.

Jika instans DB yang ingin Anda hapus memiliki replika baca, Anda harus mempromosikan atau menghapus replika baca tersebut.

Pada contoh berikut, Anda menghapus instans DB dengan dan tanpa snapshot DB akhir.

Menghapus instans DB dengan Tanpa Snapshot Akhir

Jika Anda ingin menghapus instans DB dengan cepat, Anda dapat melewati pembuatan snapshot DB akhir. Ketika Anda menghapus sebuah klaster, semua backup otomatis tersebut dihapus dan tidak dapat dipulihkan. Snapshot manual tidak dihapus.

Untuk menghapus instans DB tanpa snapshot DB akhir menggunakan konsol Neptune

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Basis data.
3. Di daftar Instans, pilih tombol radio di samping instans DB yang ingin Anda hapus.
4. Pilih Tindakan Instans, lalu pilih Hapus.
5. Pilih Tidak di kotak Membuat snapshot akhir?.
6. Pilih Hapus.

Menghapus Instans DB dengan Snapshot Akhir

Jika Anda ingin memulihkan instans DB Anda yang dihapus nanti, buat snapshot DB akhir. Semua backup otomatis juga dihapus dan tidak dapat dipulihkan. Snapshot manual tidak dihapus.

Untuk menghapus instans DB dengan snapshot DB akhir menggunakan konsol Neptune

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Basis data.
3. Di daftar Instans, pilih tombol radio di samping instans DB yang ingin Anda hapus.
4. Pilih Tindakan Instans, lalu pilih Hapus.
5. Pilih Ya di kotak Membuat snapshot akhir?.
6. Di kotak Nama snapshot akhir, masukkan nama snapshot DB akhir Anda.
7. Pilih Hapus.

Anda dapat memeriksa kesehatan sebuah instance, menentukan jenis instans, mencari tahu versi rilis mesin yang saat ini telah Anda instal, dan mendapatkan informasi lain tentang sebuah instance menggunakan API status instance.

Menangkap perubahan grafik secara real time menggunakan aliran Neptunus

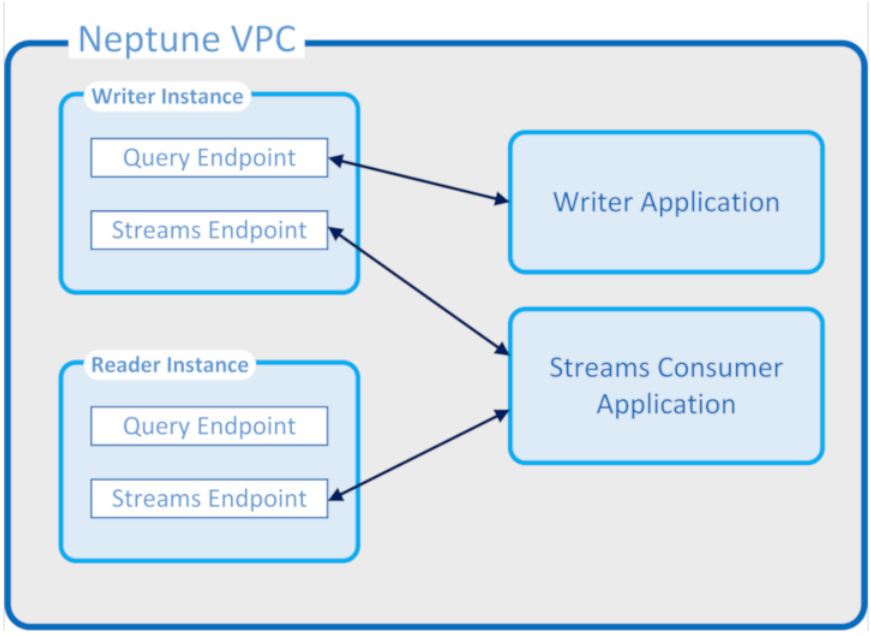
Pengaliran Neptune mencatat setiap perubahan di grafik Anda seperti yang terjadi, dalam urutan yang dibuat, dengan cara yang terkelola sepenuhnya. Setelah Anda mengaktifkan Stream, Neptune mengurus ketersediaan, cadangan, keamanan dan kedaluwarsa.

Berikut ini adalah beberapa dari banyak kasus penggunaan di mana Anda mungkin ingin menangkap perubahan pada grafik yang terjadi:

- Anda mungkin ingin aplikasi Anda memberi tahu orang secara otomatis saat perubahan tertentu dibuat.
- Anda mungkin ingin mempertahankan versi data grafik saat ini di penyimpanan data lain juga, seperti Amazon OpenSearch Service, Amazon ElastiCache, atau Amazon Simple Storage Service (Amazon S3).

Neptune menggunakan penyimpanan asli yang sama untuk pengaliran perubahan-log seperti untuk data grafik. Neptune menulis entri log perubahan secara sinkron bersama-sama dengan transaksi yang membuat perubahan tersebut. Anda mengambil catatan perubahan ini dari pengaliran log menggunakan HTTP REST API. (Untuk informasi, lihat Memanggil Streams API.)

Diagram berikut menunjukkan cara data perubahan-log dapat diambil dari Pengaliran Neptune.



Jaminan aliran Neptunus

- Perubahan yang dilakukan oleh transaksi segera tersedia untuk pembacaan dari penulis dan pembaca segera setelah transaksi selesai (selain dari setiap lag replikasi normal dalam pembaca).
- Catatan perubahan muncul secara ketat berurutan, dalam urutan di mana perubahan terjadi (ini termasuk perubahan yang dilakukan dalam transaksi).
- Pengaliran perubahan tidak mengandung duplikat. Setiap perubahan dicatat hanya sekali.
- Pengaliran perubahan selesai. Tidak ada perubahan yang hilang atau dihilangkan.
- Pengaliran perubahan berisi semua informasi yang diperlukan untuk menentukan keadaan lengkap database itu sendiri pada setiap titik waktu, asalkan keadaan awal diketahui.
- Pengaliran dapat diaktifkan atau dinonaktifkan kapan saja.

Neptunus mengalirkan properti operasional

- Pengaliran perubahan-log sepenuhnya dikelola.
- Data perubahan-log ditulis serentak sebagai bagian dari transaksi yang sama yang membuat perubahan.
- Saat Neptunus Streams diaktifkan, Anda I/O dikenakan dan biaya penyimpanan yang terkait dengan data log perubahan.
- Secara default, catatan perubahan secara otomatis dibersihkan satu minggu setelah dibuat. Dimulai dengan rilis mesin 1.2.0.0, periode retensi ini dapat diubah menggunakan parameter cluster DB neptune_streams_expiry_days ke sejumlah hari antara 1 dan 90.
- Baca performa pada skala pengaliran dengan instans.
- Anda dapat mencapai ketersediaan tinggi dan membaca throughput menggunakan replika baca. Tidak ada batas pada jumlah pembaca pengaliran yang dapat Anda buat dan gunakan secara bersamaan.
- Data perubahan-log direplikasi di beseluruh Availability Zones, membuatnya sangat tahan lama.
- Data log seaman data grafik Anda sendiri. Ini dapat dienkripsi saat istirahat dan saat transit. Akses dapat dikontrol menggunakan IAM, Amazon VPC, AWS Key Management Service dan AWS KMS(). Seperti data grafik, dapat dicadangkan dan kemudian dipulihkan menggunakan point-in-time restores (PITR).
- Penulisan sinkron data pengaliran sebagai bagian dari setiap transaksi menyebabkan sedikit degradasi dalam performa tulis secara keseluruhan.
- Data pengaliran tidak dipecah, karena Neptune adalah single-sharded sesuai desain.

- Pengaliran log API `GetRecords` menggunakan sumber daya yang sama seperti semua operasi grafik Neptune lainnya. Ini berarti bahwa klien perlu melakukan load balance antara permintaan pengaliran dan permintaan DB lainnya.
- Ketika pengaliran dinonaktifkan, semua data log segera menjadi tidak dapat diakses. Ini berarti bahwa Anda harus membaca semua data log yang menarik bagi Anda sebelum Anda menonaktifkan logging.
- Saat ini tidak ada integrasi asli dengan AWS Lambda. Pengaliran log tidak menghasilkan peristiwa yang dapat memicu fungsi Lambda.

Topik

- Menggunakan Neptune Streams
- Format Serialisasi dalam Neptune Stream
- Contoh Streams Neptune
- Menggunakan AWS CloudFormation untuk Mengatur Neptune-to-Neptune Replikasi dengan Aplikasi Konsumen Streams
- Menggunakan aliran Neptunus replikasi lintas wilayah untuk pemulihan bencana

Menggunakan Neptune Streams

Dengan fitur Neptune Streams, Anda dapat membuat urutan lengkap entri log perubahan yang merekam setiap perubahan yang dibuat pada data grafik Anda saat itu terjadi. Untuk gambaran umum fitur ini, lihat Menangkap perubahan grafik secara real time menggunakan aliran Neptunus.

Topik

- Mengaktifkan Neptune Streams
- Menonaktifkan Neptune Streams
- Memanggil Neptune Streams REST API
- Format Respon Neptune Streaming API
- Pengecualian Neptune Streams API

Mengaktifkan Neptune Streams

Anda dapat mengaktifkan atau menonaktifkan Aliran Neptunus kapan saja dengan mengatur `neptune_streams` parameter cluster DB. Mengatur parameter ke 1 mengaktifkan Stream dan mengaturnya ke `0` menonaktifkan Streams.

> **Note**
>
> Setelah mengubah parameter cluster `neptune_streams` DB, Anda harus me-reboot semua instance DB di cluster agar perubahan diterapkan.

Anda dapat mengatur parameter cluster DB neptune_streams_expiry_days untuk mengontrol berapa hari, dari 1 hingga 90, bahwa catatan aliran tetap ada di server sebelum dihapus. Defaultnya adalah 7.

Neptune Streams awalnya diperkenalkan sebagai fitur eksperimental yang Anda aktifkan atau nonaktifkan dalam Mode Lab menggunakan parameter DB `neptune_lab_mode` Cluster (lihat). Mode Lab Neptune Menggunakan Mode Lab untuk mengaktifkan Stream sekarang tidak berlaku lagi dan akan dinonaktifkan di masa mendatang.

Menonaktifkan Neptune Streams

Anda dapat menonaktifkan Neptune Stream yang berproses kapan saja.

Untuk menonaktifkan Streaming, perbarui grup parameter Cluster DB sehingga nilai parameter `neptune_streams` diatur ke 0.

> **Important**
>
> Segera setelah Streams dimatikan, Anda tidak dapat mengakses data perubahan-log lagi. Pastikan untuk membaca apa yang Anda minati di sebelum mematikan Streams.

Memanggil Neptune Streams REST API

Anda mengakses Neptune Streaming menggunakan REST API yang mengirimkan permintaan HTTP GET ke salah satu titik akhir lokal berikut:

- Untuk DB grafik SPARQL: `https://`*`Neptune-DNS`*`:8182/sparql/stream`.

- Untuk grafik Gremlin atau OpenCypher DB: atau. `https://Neptune-DNS:8182/propertygraph/stream` `https://Neptune-DNS:8182/pg/stream`

> **Note**
>
> Titik akhir aliran Gremlin (`https://Neptune-DNS:8182/gremlin/stream`) tidak digunakan lagi, bersama dengan format keluaran terkait (). `GREMLIN_JSON` Ini masih didukung untuk kompatibilitas mundur tetapi dapat dihapus di rilis mendatang.

Hanya HTTP operasi `GET` diperbolehkan.

Neptune mendukung kompresi `gzip` respon, asalkan permintaan HTTP mencakup Header `Accept-Encoding` yang menentukan `gzip` sebagai format kompresi yang diterima (yaitu, `"Accept-Encoding: gzip"`).

Parameter

- `limit`- panjang, opsional. Kisaran: 1—100.000. Default: 10.

  Menentukan angka maksimum catatan yang akan dikembalikan. Ada juga batas ukuran 10 MB pada respon yang tidak dapat diubah dan yang diutamakan daripada jumlah catatan yang ditentukan dalam parameter `limit`. Respon mencakup catatan pelanggaran ambang batas jika batas 10 MB tercapai.
- `iteratorType`— String, opsional.

  Parameter ini dapat berupa salah satu nilai berikut:
  - `AT_SEQUENCE_NUMBER`(default) - Menunjukkan bahwa pembacaan harus dimulai dari nomor urutan acara yang ditentukan bersama oleh `opNum` parameter `commitNum` dan.
  - `AFTER_SEQUENCE_NUMBER`— Menunjukkan bahwa pembacaan harus dimulai tepat setelah nomor urut acara yang ditentukan bersama oleh `opNum` parameter `commitNum` dan.
  - `TRIM_HORIZON`— Menunjukkan bahwa pembacaan harus dimulai pada catatan terakhir yang belum dipangkas dalam sistem, yang merupakan catatan tertua yang belum kedaluwarsa (belum dihapus) dalam aliran log perubahan. Mode ini berguna selama startup aplikasi, ketika Anda tidak memiliki nomor urutan acara awal tertentu.
  - `LATEST`— Menunjukkan bahwa pembacaan harus dimulai pada catatan terbaru dalam sistem, yang merupakan catatan terbaru yang belum kedaluwarsa (belum dihapus) dalam aliran log

perubahan. Ini berguna ketika ada kebutuhan untuk membaca catatan dari bagian atas aliran saat ini agar tidak memproses catatan yang lebih lama, seperti selama pemulihan bencana atau peningkatan zero-downtime. Perhatikan bahwa dalam mode ini, paling banyak hanya ada satu catatan yang dikembalikan.

- `commitNum`— panjang, diperlukan ketika `AT_SEQUENCE_NUMBER` IteratorType adalah atau. `AFTER_SEQUENCE_NUMBER`

  Jumlah commit dari catatan awal yang akan dibaca dari stream log perubahan.

  Parameter ini diabaikan ketika `iteratorType` adalah `TRIM_HORIZON` atau`LATEST`.

- `opNum`— panjang, opsional (defaultnya adalah`1`).

  Nomor urut operasi dalam commit yang ditentukan untuk mulai dibaca dari dalam data stream log perubahan.

Operasi yang mengubah data grafik SPARQL umumnya hanya menghasilkan catatan perubahan tunggal per operasi. Namun, operasi yang mengubah data grafik Gremlin dapat menghasilkan beberapa catatan perubahan per operasi, seperti dalam contoh berikut:

- `INSERT`— Sebuah simpul Gremlin dapat memiliki beberapa label, dan elemen Gremlin dapat memiliki beberapa properti. Sebuah catatan perubahan terpisah yang dihasilkan untuk setiap label dan properti ketika elemen dimasukkan.
- `UPDATE`— Ketika properti elemen Gremlin diubah, dua catatan perubahan dihasilkan: yang pertama untuk menghapus nilai sebelumnya, dan yang kedua untuk memasukkan nilai baru.
- `DELETE`— Catatan perubahan terpisah dihasilkan untuk setiap properti elemen yang dihapus. Misalnya, ketika edge Gremlin dengan properti dihapus, satu catatan perubahan dihasilkan untuk masing-masing properti, dan setelah itu, satu dihasilkan untuk penghapusan label edge.

  Ketika vertex Gremlin dihapus, semua properti edge masuk dan keluar dihapus terlebih dahulu, selanjutnya label edge, selanjutnya properti vertex, dan terakhir label vertex. Setiap penghapusan ini menghasilkan catatan perubahan.

Format Respon Neptune Streaming API

Respon terhadap permintaan Neptune Streaming REST API memiliki bidang-bidang berikut:

- `lastEventId`— Pengidentifikasi urutan dari perubahan terakhir dalam respons aliran. ID peristiwa disusun dari dua bidang: A `commitNum` mengidentifikasi transaksi yang mengubah grafik, dan `opNum` mengidentifikasi operasi tertentu dalam transaksi itu. Ini ditunjukkan dalam contoh berikut.

```
  "eventId": {
    "commitNum": 12,
    "opNum": 1
  }
```

- `lastTrxTimestamp`— Waktu di mana komit untuk transaksi diminta, dalam milidetik dari zaman Unix.
- `format`— Format serialisasi untuk catatan perubahan yang dikembalikan. Nilai yang mungkin adalah `PG_JSON` untuk catatan perubahan Gremlin atau OpenCypher, dan `NQUADS` untuk catatan perubahan SPARQL.
- `records`— Array catatan aliran log perubahan serial yang disertakan dalam respons. Setiap record dalam `records` array berisi bidang-bidang ini:
  - `commitTimestamp`— Waktu di mana komit untuk transaksi diminta, dalam milidetik dari zaman Unix.
  - `eventId`— Pengidentifikasi urutan catatan perubahan aliran.
  - `data`— Serial Gremlin, SPARQL, atau catatan perubahan. OpenCypher Format serialisasi setiap catatan dijelaskan secara lebih rinci di bagian selanjutnya,Format Serialisasi dalam Neptune Stream.
  - op— Operasi yang menciptakan perubahan.
  - `isLastOp`— Hanya hadir jika operasi ini adalah yang terakhir dalam transaksinya. Saat hadir, itu diatur ke`true`. Berguna untuk memastikan bahwa seluruh transaksi dikonsumsi.
- `totalRecords`— Jumlah total catatan dalam tanggapan.

Misalnya, respons berikut mengembalikan data perubahan Gremlin, untuk transaksi yang berisi lebih dari satu operasi:

```
{
  "lastEventId": {
    "commitNum": 12,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
```

```
  "records": [
    {
      "commitTimestamp": 1560011610678,
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "ADD"
    }
  ],
  "totalRecords": 1
}
```

Respons berikut mengembalikan data perubahan SPARQL untuk operasi terakhir dalam transaksi (operasi diidentifikasi oleh `EventId(97, 1)` dalam nomor transaksi 97).

```
{
  "lastEventId": {
    "commitNum": 97,
    "opNum": 1
  },
  "lastTrxTimestamp": 1561489355102,
  "format": "NQUADS",
  "records": [
    {
      "commitTimestamp": 1561489355102,
      "eventId": {
        "commitNum": 97,
        "opNum": 1
      },
      "data": {
        "stmt": "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
      },
      "op": "ADD",
```

```
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Pengecualian Neptune Streams API

Tabel berikut menjelaskan pengecualian Neptune Streams.

Kode Kesalahan	Kode HTTP	OK atau Mengulang?	Pesan
`InvalidParameterException`	400	Tidak	out-of-rangeNilai atau tidak valid diberikan sebagai parameter input.
`ExpiredStreamException`	400	Tidak	Semua catatan yang diminta melebihi usia maksimum yang diizinkan dan telah kedaluwarsa.
`ThrottlingException`	500	Ya	Tingkat permintaan melebihi throughput maksimum.
`StreamRecordsNotFoundException`	404	Tidak	Sumber daya yang diminta tidak dapat ditemukan. Stream mungkin tidak ditentukan dengan benar.
`MemoryLimitExceededException`	500	Ya	Pemrosesan permintaan tidak berhasil karena kurangnya memori, tetapi dapat dicoba

Kode Kesalahan	Kode HTTP	OK atau Mengulang?	Pesan
			lagi ketika server tidak terlalu sibuk.

Format Serialisasi dalam Neptune Stream

Amazon Neptune menggunakan dua format yang berbeda untuk serialisasi data grafik-perubahan untuk stream log, tergantung apakah grafik dibuat menggunakan Gremlin atau SPARQL.

Kedua format berbagi format serialisasi rekaman umum, seperti yang dijelaskan dalamFormat Respon Neptune Streaming API, yang berisi bidang berikut:

- `commitTimestamp`— Waktu di mana komit untuk transaksi diminta, dalam milidetik dari zaman Unix.
- `eventId`— Pengidentifikasi urutan catatan perubahan aliran.
- `data`— Serial Gremlin, SPARQL, atau catatan perubahan. OpenCypher Format serialisasi setiap catatan dijelaskan secara lebih rinci di bagian selanjutnya.
- `op`— Operasi yang menciptakan perubahan.

Topik

- PG_JSON Ubah Format Serialisasi
- Format Serialisasi Perubahan SPARQL NQUADS

PG_JSON Ubah Format Serialisasi

> **Note**
>
> Output Gremlin stream output format (`GREMLIN_JSON`) oleh Gremlin stream endpoint () tidak digunakan lagi. `https://Neptune-DNS:8182/gremlin/stream` Ini digantikan oleh PG_JSON, yang saat ini identik dengan. `GREMLIN_JSON`

Catatan perubahan Gremlin atau OpenCypher, yang terdapat di `data` bidang respons aliran log, memiliki bidang berikut:

- `id` – STRING - Diperlukan.

  ID dari elemen Gremlin atau OpenCypher.

- `type` – STRING - Diperlukan.

  Jenis elemen Gremlin atau OpenCypher ini. Harus berupa salah satu dari yang berikut:

  - `vl`— Label Vertex untuk Gremlin; label node untuk OpenCypher.
  - `vp`- Properti Vertex untuk Gremlin; properti node untuk OpenCypher.
  - `e`- Label tepi dan tepi untuk Gremlin; jenis hubungan dan hubungan untuk OpenCypher.
  - `ep`- Properti tepi untuk Gremlin; properti hubungan untuk OpenCypher.

- `key` – STRING - Diperlukan.

  Nama properti. Untuk label elemen, ini adalah “label”.

- `value`— objek `value`, diperlukan.

  Ini adalah objek JSON yang berisi bidang `value` untuk nilai itu sendiri, dan bidang `datatype` untuk tipe data JSON dari nilai tersebut.

  ```
  "value": {
    "value": "the new value",
    "dataType": "the JSON datatype of the new value"
  }
  ```

- `from`- String, opsional.

  Jika ini adalah tepi (type="e”), ID yang sesuai dari simpul atau sumber simpul.

- `to`- String, opsional.

  Jika ini adalah tepi (type="e”), ID yang sesuai dengan simpul atau target node.

Contoh Gremlin

- Berikut ini adalah contoh label vertex Gremlin.

  ```
  {
    "id": "an ID string",
    "type": "vl",
    "key": "label",
    "value": {
  ```

```
    "value": "the new value of the vertex label",
    "dataType": "String"
  }
}
```

- Berikut ini adalah contoh properti vertex Gremlin.

```
{
  "id": "an ID string",
  "type": "vp",
  "key": "the property name",
  "value": {
    "value": "the new value of the vertex property",
    "dataType": "the datatype of the vertex property"
  }
}
```

- Berikut ini adalah contoh edge Gremlin.

```
{
  "id": "an ID string",
  "type": "e",
  "key": "label",
  "value": {
    "value": "the new value of the edge",
    "dataType": "String"
  },
  "from": "the ID of the corresponding "from" vertex",
  "to": "the ID of the corresponding "to" vertex"
}
```

Contoh OpenCypher

- Berikut ini adalah contoh label node OpenCypher.

```
{
  "id": "an ID string",
  "type": "vl",
  "key": "label",
  "value": {
    "value": "the new value of the node label",
    "dataType": "String"
```

```
  }
}
```

- Berikut ini adalah contoh dari properti node OpenCypher.

```
{
  "id": "an ID string",
  "type": "vp",
  "key": "the property name",
  "value": {
    "value": "the new value of the node property",
    "dataType": "the datatype of the node property"
  }
}
```

- Berikut ini adalah contoh hubungan OpenCypher.

```
{
  "id": "an ID string",
  "type": "e",
  "key": "label",
  "value": {
    "value": "the new value of the relationship",
    "dataType": "String"
  },
  "from": "the ID of the corresponding source node",
  "to": "the ID of the corresponding target node"
}
```

Format Serialisasi Perubahan SPARQL NQUADS

Log Neptune berubah ke quad SPARQL dalam grafik menggunakan bahasa `N-QUADS` Resource Description Framework (RDF) yang didefinisikan dalam spesifikasi W3C RDF 1.1 N-Quad.

Bidang `data` dalam catatan perubahan hanya berisi bidang `stmt` yang memegang pernyataan N-QUADS mengekspresikan quad yang diubah, seperti dalam contoh berikut.

```
  "stmt" : "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
```

Contoh Streams Neptune

Contoh berikut menunjukkan cara mengakses data streams perubahan-log di Amazon Neptune.

Topik

- AT_SEQUENCE_NUMBERUbah Log
- AFTER_SEQUENCE_NUMBERUbah Log
- TRIM_HORIZONUbah Log
- LATESTUbah Log
- Log Perubahan Kompresi

AT_SEQUENCE_NUMBER Ubah Log

Contoh berikut menunjukkan log perubahan Gremlin atau OpenCypherAT_SEQUENCE_NUMBER.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AT_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
    {
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "commitTimestamp": 1560011610678,
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
```

```
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Yang ini menunjukkan contoh SPARQL dari log AT_SEQUENCE_NUMBER perubahan.

```
curl -s "https://localhost:8182/sparql/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AT_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1571252030566,
  "format": "NQUADS",
  "records": [
    {
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "commitTimestamp": 1571252030566,
      "data": {
        "stmt": "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

AFTER_SEQUENCE_NUMBER Ubah Log

Contoh berikut menunjukkan log perubahan Gremlin atau OpenCypherAFTER_SEQUENCE_NUMBER.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AFTER_SEQUENCE_NUMBER" |jq
{
```

```
  "lastEventId": {
    "commitNum": 2,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011633768,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1560011633768,
      "eventId": {
        "commitNum": 2,
        "opNum": 1
      },
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "REMOVE",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

TRIM_HORIZON Ubah Log

Contoh berikut menunjukkan log perubahan Gremlin atau OpenCypher`TRIM_HORIZON`.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&iteratorType=TRIM_HORIZON" |jq
{
  "lastEventId": {
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
```

```
    {
      "commitTimestamp": 1560011610678,
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

LATESTUbah Log

Contoh berikut menunjukkan log perubahan Gremlin atau OpenCypherLATEST. Perhatikan bahwa parameter API`limit`,`commitNum`, dan opNum sepenuhnya opsional.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?iteratorType=LATEST" | jq
{
  "lastEventId": {
    "commitNum": 21,
    "opNum": 4
  },
  "lastTrxTimestamp": 1634710497743,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1634710497743,
      "eventId": {
        "commitNum": 21,
        "opNum": 4
      },
      "data": {
```

```
        "id": "24be4e2b-53b9-b195-56ba-3f48fa2b60ac",
        "type": "e",
        "key": "label",
        "value": {
          "value": "created",
          "dataType": "String"
        },
        "from": "4",
        "to": "5"
      },
      "op": "REMOVE",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Log Perubahan Kompresi

Contoh berikut menunjukkan log perubahan kompresi Gremlin atau OpenCypher.

```
curl -sH \
  "Accept-Encoding: gzip" \
  "https://Neptune-DNS:8182/propertygraph/stream?limit=1&commitNum=1" \
  -H "Accept-Encoding: gzip" \
  -v |gunzip -|jq
> GET /propertygraph/stream?limit=1 HTTP/1.1
> Host: localhost:8182
> User-Agent: curl/7.64.0
> Accept: /
> Accept-Encoding: gzip
*> Accept-Encoding: gzip*
>
< HTTP/1.1 200 OK
< Content-Type: application/json; charset=UTF-8
< Connection: keep-alive
*< content-encoding: gzip*
< content-length: 191
<
{ [191 bytes data]
Connection #0 to host localhost left intact
{
  "lastEventId": "1:1",
```

```
  "lastTrxTimestamp": 1558942160603,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1558942160603,
      "eventId": "1:1",
      "data": {
        "id": "v1",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "person",
          "dataType": "String"
        }
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Menggunakan AWS CloudFormation untuk Mengatur Neptune-to-Neptune Replikasi dengan Aplikasi Konsumen Streams

Anda dapat menggunakan CloudFormation template untuk menyiapkan aplikasi konsumen streaming Neptunus untuk mendukung replikasi. Neptune-to-Neptune

Topik

- Pilih CloudFormation template untuk Wilayah Anda
- Tambahkan detail Tentang tumpukan konsumen streaming Neptunus yang Anda buat
- Jalankan CloudFormation Template
- Untuk memperbarui poller aliran dengan artefak Lambda terbaru

Pilih CloudFormation template untuk Wilayah Anda

Untuk meluncurkan CloudFormation tumpukan yang sesuai di CloudFormation konsol, pilih salah satu tombol Launch Stack di tabel berikut, tergantung pada AWS Wilayah yang ingin Anda gunakan.

Region	Lihat	Lihat di Designer	Luncurkan
US East (N. Virginia)	Lihat	Lihat di Desainer	
AS Timur (Ohio)	Lihat	Lihat di Desainer	
US West (N. California)	Lihat	Lihat di Desainer	
US West (Oregon)	Lihat	Lihat di Desainer	
Kanada (Pusat)	Lihat	Lihat di Desainer	
Amerika Selatan (Sao Paulo)	Lihat	Lihat di Desainer	
Europe (Stockholm)	Lihat	Lihat di Desainer	
Eropa (Irlandia)	Lihat	Lihat di Desainer	
Eropa (London)	Lihat	Lihat di Desainer	
Europe (Paris)	Lihat	Lihat di Desainer	
Eropa (Frankfurt)	Lihat	Lihat di Desainer	
Timur Tengah (Bahrain)	Lihat	Lihat di Desainer	
Timur Tengah (UAE)	Lihat	Lihat di Desainer	
Israel (Tel Aviv)	Lihat	Lihat di Desainer	

Region	Lihat	Lihat di Designer	Luncurkan
Africa (Cape Town)	Lihat	Lihat di Desainer	
Asia Pacific (Tokyo)	Lihat	Lihat di Desainer	
Asia Pasifik (Hong Kong)	Lihat	Lihat di Desainer	
Asia Pasifik (Seoul)	Lihat	Lihat di Desainer	
Asia Pacific (Singapore)	Lihat	Lihat di Desainer	
Asia Pacific (Sydney)	Lihat	Lihat di Desainer	
Asia Pasifik (Mumbai)	Lihat	Lihat di Desainer	
Tiongkok (Beijing)	Lihat	Lihat di Desainer	
Tiongkok (Ningxia)	Lihat	Lihat di Desainer	
AWS GovCloud (AS-Barat)	Lihat	Lihat di Desainer	
AWS GovCloud (AS-Timur)	Lihat	Lihat di Desainer	

Pada halaman Buat Tumpukan, pilih Selanjutnya.

Tambahkan detail Tentang tumpukan konsumen streaming Neptunus yang Anda buat

Halaman Tentukan Detail Tumpukan menyediakan properti dan parameter yang dapat Anda gunakan untuk mengontrol pengaturan aplikasi:

Nama Stack — Nama CloudFormation tumpukan baru yang Anda buat. Anda dapat menggunakan nilai default secara umum, `NeptuneStreamPoller`.

Di bawah Parameter, berikan yang berikut ini:

Konfigurasi jaringan untuk VPC Dimana konsumen streaming berjalan

- **`VPC`**— Berikan nama VPC tempat fungsi Lambda polling akan berjalan.
- **`SubnetIDs`**— Subnet tempat antarmuka jaringan didirikan. Tambahkan subnet yang sesuai dengan klaster Neptune Anda.
- **`SecurityGroupIds`**— Berikan grup keamanan IDs yang memberikan akses masuk tulis ke cluster DB Neptunus sumber Anda.
- **`RouteTableIds`**— Ini diperlukan untuk membuat titik akhir Amazon DynamoDB di VPC Neptunus Anda, jika Anda belum memilikinya. Anda harus memberikan daftar tabel rute yang dipisahkan koma yang IDs terkait dengan subnet.
- **`CreateDDBVPCEndPoint`**— Nilai Boolean yang defaultnya`true`, menunjukkan apakah perlu membuat titik akhir VPC Dynamo DB atau tidak. Anda hanya perlu mengubahnya menjadi `false` jika Anda telah membuat titik akhir DynamoDB di VPC Anda.
- **`CreateMonitoringEndPoint`**— Nilai Boolean yang defaultnya`true`, menunjukkan apakah perlu membuat titik akhir VPC pemantauan atau tidak.. Anda hanya perlu mengubahnya menjadi `false` jika Anda telah membuat titik akhir pemantauan di VPC Anda.

Poller Stream

- **`ApplicationName`**— Anda biasanya dapat meninggalkan set ini ke default (`NeptuneStream`). Jika Anda menggunakan nama yang berbeda, itu harus unik.
- **`LambdaMemorySize`**— Digunakan untuk mengatur ukuran memori yang tersedia untuk fungsi poller Lambda. Nilai default-nya adalah 2,048 megabyte.
- **`LambdaRuntime`**— Bahasa yang digunakan dalam fungsi Lambda yang mengambil item dari aliran Neptunus. Anda dapat mengatur ini baik ke `python3.9` atau ke `java8`.
- **`LambdaS3Bucket`**- Bucket Amazon S3 yang berisi artefak kode Lambda. Biarkan ini kosong kecuali Anda menggunakan fungsi polling Lambda khusus yang dimuat dari bucket Amazon S3 yang berbeda.
- **`LambdaS3Key`**— Kunci Amazon S3 yang sesuai dengan artefak kode Lambda Anda. Biarkan ini kosong kecuali Anda menggunakan fungsi polling Lambda khusus.
- **`LambdaLoggingLevel`**— Secara umum, biarkan set ini ke nilai default, yaitu`INFO`.

- **`ManagedPolicies`**— Daftar kebijakan terkelola yang akan digunakan untuk eksekusi fungsi Lambda Anda. Secara umum, biarkan ini kosong kecuali Anda menggunakan fungsi polling Lambda khusus.
- **`StreamRecordsHandler`**— Secara umum, biarkan kosong ini kecuali Anda menggunakan penangan khusus untuk catatan di aliran Neptunus.
- **`StreamRecordsBatchSize`**— Jumlah maksimum catatan yang akan diambil dari aliran. Anda dapat menggunakan parameter ini untuk menyetel performa. Default (`5000`) adalah tempat yang baik untuk memulai. Maksimum yang diijinkan adalah 10.000. Semakin tinggi jumlahnya, semakin sedikit panggilan jaringan yang diperlukan untuk membaca catatan dari stream, tetapi semakin banyak memori diperlukan untuk memproses catatan. Nilai yang lebih rendah dari parameter ini menghasilkan throughput yang lebih rendah.
- **`MaxPollingWaitTime`**— Waktu tunggu maksimum antara dua jajak pendapat (dalam detik). Menentukan seberapa sering poller Lambda dipanggil untuk polling aliran Neptunus. Tetapkan nilai ini ke 0 untuk polling berkelanjutan. Nilai maksimum adalah 3.600 detik (1 jam). Nilai default (60 detik) adalah tempat yang baik untuk memulai, tergantung seberapa cepat data grafik Anda berubah.
- **`MaxPollingInterval`**— Periode polling berkelanjutan maksimum (dalam hitungan detik). Gunakan ini untuk mengatur batas waktu untuk fungsi polling Lambda. Nilainya harus berada dalam kisaran antara 5 detik dan 900 detik. Nilai default (600 detik) adalah tempat yang baik untuk memulai.
- **`StepFunctionFallbackPeriod`**— Jumlah unit step-function-fallback-period untuk menunggu poller, setelah itu fungsi langkah dipanggil melalui Amazon CloudWatch Events untuk pulih dari kegagalan. Default (5 menit) adalah tempat yang baik untuk memulai.
- **`StepFunctionFallbackPeriodUnit`**— Satuan waktu yang digunakan untuk mengukur sebelumnya `StepFunctionFallbackPeriodUnit` (`minutes`,`hours`, atau`days`). Default (`minutes`) umumnya cukup.
- **`StartingCheckpoint`**— Pos pemeriksaan awal untuk poller aliran. Defaultnya adalah`0:0`, yang menandakan mulai dari awal aliran Neptunus.
- **`StreamPollerInitialState`**- Keadaan awal poller. Defaultnya adalah`ENABLED`, yang berarti replikasi aliran akan dimulai segera setelah seluruh pembuatan tumpukan selesai.

Aliran Neptunus

- **`NeptuneStreamEndpoint`**— (Wajib) Titik akhir aliran sumber Neptunus. Ini mengambil salah satu dari dua bentuk:

  - **`https://your DB cluster:port/propertygraph/stream`**(atau aliasnya,`https://your DB cluster:port/pg/stream`).
  - **`https://your DB cluster:port/sparql/stream`**.
- **`Neptune Query Engine`**— Pilih Gremlin, OpenCypher, atau SPARQL.
- **`IAMAuthEnabledOnSourceStream`**— Jika cluster DB Neptunus Anda menggunakan otentikasi IAM, atur parameter ini ke. `true`
- **`StreamDBClusterResourceId`**— Jika cluster DB Neptunus Anda menggunakan otentikasi IAM, atur parameter ini ke ID sumber daya cluster. ID sumber daya tidak sama dengan ID cluster. Sebaliknya, ia mengambil bentuk: `cluster-` diikuti oleh 28 karakter alfa-numerik. Ini dapat ditemukan di bawah Detail Klaster di konsol Neptune.

Target kluster DB Neptunus

- **`TargetNeptuneClusterEndpoint`**— Titik akhir cluster (hanya nama host) dari cluster cadangan target.

  Perhatikan bahwa jika Anda menentukan`TargetNeptuneClusterEndpoint`, Anda juga tidak dapat menentukan`TargetSPARQLUpdateEndpoint`.
- **`TargetNeptuneClusterPort`**— Nomor port untuk cluster target.

  Perhatikan bahwa jika Anda menentukan`TargetSPARQLUpdateEndpoint`, pengaturan `TargetNeptuneClusterPort` untuk diabaikan.
- **`IAMAuthEnabledOnTargetCluster`**— Setel ke true jika otentikasi IAM akan diaktifkan pada cluster target.
- **`TargetAWSRegion`**— AWS Wilayah cluster cadangan target, sepertius`-east-1`). Anda harus memberikan parameter ini hanya ketika AWS wilayah cluster cadangan target berbeda dari wilayah cluster sumber Neptunus, seperti dalam kasus replikasi lintas wilayah. Jika wilayah sumber dan target sama, parameter ini opsional.

  Perhatikan bahwa jika `TargetAWSRegion` nilainya bukan AWS wilayah valid yang didukung Neptunus, prosesnya gagal.
- **`TargetNeptuneDBClusterResourceId`**— Opsional: ini hanya diperlukan ketika otentikasi IAM diaktifkan pada cluster DB target. Setel ke ID sumber daya dari cluster target.

- **SPARQLTripleOnlyMode**— Bendera Boolean yang menentukan apakah mode triple-only diaktifkan. Dalam mode triple-only, tidak ada replikasi grafik bernama. Nilai default-nya adalah `false`.
- **TargetSPARQLUpdateEndpoint**— URL titik akhir target untuk pembaruan SPARQL, seperti. `https://abc.com/xyz` Endpoint ini dapat berupa toko SPARQL yang mendukung quad atau triple.

  Perhatikan bahwa jika Anda menentukan`TargetSPARQLUpdateEndpoint`, Anda juga tidak dapat menentukan`TargetNeptuneClusterEndpoint`, dan pengaturan diabaikan. `TargetNeptuneClusterPort`
- **BlockSparqlReplicationOnBlankNode** — Bendera Boolean yang, jika disetel ke`true`, menghentikan replikasi untuk BlankNode data SPARQL (RDF). Nilai bawaannya adalah `false`.

Alarm

- **Required to create Cloud watch Alarm**— Atur ini ke `true` jika Anda ingin membuat CloudWatch alarm untuk tumpukan baru.
- **SNS Topic ARN for Cloudwatch Alarm Notifications**— Topik SNS ARN CloudWatch tempat pemberitahuan alarm harus dikirim (hanya diperlukan jika alarm diaktifkan).
- **Email for Alarm Notifications**— Alamat email tempat pemberitahuan alarm harus dikirim (hanya diperlukan jika alarm diaktifkan).

Untuk tujuan notifikasi alarm, Anda dapat menambahkan SNS saja, hanya email, atau SNS dan email.

Jalankan CloudFormation Template

Sekarang Anda dapat menyelesaikan proses instans aplikasi konsumen stream Neptune provisioning sebagai berikut:

1. Di CloudFormation, pada halaman Tentukan Detail Tumpukan, pilih Berikutnya.
2. Pada halaman Opsi, pilih Selanjutnya.
3. Pada halaman Review, pilih kotak centang pertama untuk mengetahui bahwa CloudFormation akan membuat sumber daya IAM. Pilih kotak centang kedua untuk mengetahui `CAPABILITY_AUTO_EXPAND` untuk tumpukan baru.

> **Note**
>
> `CAPABILITY_AUTO_EXPAND` secara eksplisit mengakui bahwa macro akan diperluas saat membuat tumpukan, tanpa review sebelumnya. Pengguna sering kali membuat perubahan yang ditetapkan dari templat yang diproses, sehingga perubahan yang dibuat oleh makro bisa direview tepat sebelum membuat tumpukan. Untuk informasi selengkapnya, lihat CloudFormation CreateStackAPI di Referensi AWS CloudFormation API.

Lalu pilih Buat.

Untuk memperbarui poller aliran dengan artefak Lambda terbaru

Anda dapat memperbarui poller aliran dengan artefak kode Lambda terbaru sebagai berikut:

1. Di Konsol Manajemen AWS, navigasikan ke CloudFormation dan pilih CloudFormation tumpukan induk utama.
2. Pilih opsi Perbarui untuk tumpukan.
3. Pilih Ganti template saat ini.
4. Untuk sumber template, pilih URL Amazon S3 dan masukkan URL S3 berikut:

```
https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
neptune_to_neptune.json
```

5. Pilih Berikutnya tanpa mengubah CloudFormation parameter apa pun.
6. Pilih Update Stack.

Tumpukan sekarang akan memperbarui artefak Lambda dengan yang terbaru.

Menggunakan aliran Neptunus replikasi lintas wilayah untuk pemulihan bencana

Neptune menyediakan dua cara untuk menerapkan kemampuan failover lintas-wilayah:

- Salinan dan pemulihan snapshot lintas-wilayah
- Menggunakan pengaliran Neptune untuk mereplikasi data antara dua klaster di dua wilayah yang berbeda.

Salinan dan pemulihan snapshot lintas-wilayah memiliki overhead operasional terendah untuk memulihkan klaster Neptune di wilayah yang berbeda. Namun, menyalin snapshot antar daerah memerlukan waktu transfer data yang signifikan, karena snapshot adalah cadangan penuh dari klaster Neptune. Hasilnya, salinan dan pemulihan snapshot lintas-wilayah dapat digunakan sebagai skenario yang hanya memerlukan Recovery Point Objective (RPO) jam dan waktu pemulihan tujuan (RTO) jam.

Recovery Point Objective (RPO) diukur dengan waktu di antara backup. Ini mendefinisikan berapa banyak data yang mungkin hilang antara waktu cadangan terakhir dibuat dan waktu di mana database dipulihkan.

Recovery Point Objective (RTO) diukur dengan waktu yang diperlukan untuk melaksanakan operasi pemulihan. Ini adalah waktu yang dibutuhkan klaster DB melakukan fail over ke database yang dipulihkan setelah terjadi kegagalan.

Pengaliran Neptune menyediakan cara untuk menjaga backup klaster Neptune sinkron dengan klaster produksi utama setiap saat. Jika terjadi kegagalan, database Anda kemudian gagal ke klaster backup. Hal ini mengurangi RPO dan RTO hingga ukuran menit, karena data terus-menerus disalin ke klaster backup, yang segera tersedia sebagai target failover setiap saat.

Kelemahan menggunakan pengaliran Neptune dengan cara ini adalah bahwa kedua overhead operasional yang diperlukan untuk mempertahankan komponen replikasi, dan biaya memiliki kedua Neptune DB klaster online sepanjang waktu, menjadi signifikan.

Menyiapkan Neptune-to-Neptune replikasi

Klaster DB produksi utama Anda berada di VPC di wilayah sumber tertentu. Ada tiga hal utama yang Anda butuhkan untuk meniru atau menyamai di wilayah pemulihan yang berbeda untuk tujuan pemulihan bencana:

- Data yang disimpan dalam klaster.
- Konfigurasi klaster utama. Ini akan mencakup apakah menggunakan otentikasi IAM, apakah itu dienkripsi, parameter klaster DB-nya, parameter instans-nya, ukuran instans, dan sebagainya).

- Topologi jaringan yang digunakan, termasuk di dalamnya target VPC, grup keamanan, dan sebagainya.

Anda dapat menggunakan APIs manajemen Neptunus seperti berikut ini untuk mengumpulkan informasi tersebut:

- `DescribeDBClusters`
- `DescribeDBInstances`
- `DescribeDBClusterParameters`
- `DescribeDBParameters`
- `DescribeVpcs`

Dengan informasi yang Anda kumpulkan, Anda dapat menggunakan prosedur berikut ini untuk mengatur claster backup di wilayah yang berbeda, saat klaster produksi Anda gagal ke peristiwa kegagalan.

Aktifkan aliran Neptunus

Anda dapat menggunakan Memodifikasi DBCluster ParameterGroup untuk mengatur parameter `neptune_streams` ke 1. Kemudian, reboot semua instans di klaster DB sehingga perubahan berlaku.

Adalah ide yang baik untuk melakukan setidaknya satu operasi tambahan atau operasi pembaruan pada klaster DB sumber setelah pengaliran Neptune telah diaktifkan. Ini mengisi pengaliran perubahan dengan titik data yang dapat direferensikan kemudian, ketika mesinkronkan ulang klaster produksi dengan cluster backup.

Buat VPC baru di wilayah tempat Anda ingin menyiapkan klaster cadangan

Sebelum membuat klaster DB Neptune baru di wilayah yang berbeda dari klaster utama Anda, Anda perlu membangun VPC baru di wilayah target untuk menjadi tuan rumah cluster. Konektivitas antara cluster primer dan cadangan dibuat melalui peering VPC, yang menggunakan lalu lintas lintas subnet pribadi secara berbeda. VPCs Namun, untuk membuat VPC mengintip di antara dua VPCs, mereka tidak boleh memiliki blok CIDR atau ruang alamat IP yang tumpang tindih. Hal ini bahwa Anda tidak bisa hanya menggunakan VPC default di kedua wilayah, karena blok CIDR untuk VPC default selalu sama (`172.31.0.0/16`).

Anda dapat menggunakan VPC yang ada di wilayah target selama memenuhi kondisi berikut:

- Ini tidak memiliki blok CIDR yang tumpang tindih dengan blok CIDR VPC tempat klaster utama Anda berada.
- Hal ini belum dipasangkan dengan VPC lain yang memiliki blok CIDR yang sama seperti VPC tempat klaster utama Anda berada.

Jika tidak ada VPC yang cocok tersedia di wilayah target, buatlah satu menggunakan Amazon EC2 `CreateVpc` API.

Buat snapshot dari cluster utama Anda dan kembalikan ke wilayah cadangan target

Sekarang Anda membuat klaster Neptune baru di VPC yang sesuai di wilayah target cadangan yang merupakan salinan klaster produksi Anda:

Membuat salinan klaster produksi Anda di wilayah cadangan

1. Di wilayah backup target Anda, buat ulang parameter dan grup parameter yang digunakan oleh klaster DB produksi Anda. Anda dapat melakukannya menggunakan `CreateDBClusterParameterGroup`, `CreateDBParameterGroup`, `ModifyDBClusterParameterGroup`, dan `ModifyDBParameterGroup`.

   Perhatikan bahwa `CopyDBClusterParameterGroup`dan saat ini `CopyDBParameterGroup` APIs tidak mendukung penyalinan lintas wilayah.
2. Gunakan `CreateDBClusterSnapshot` untuk membuat snapshot dari klaster produksi Anda di VPC di wilayah produksi Anda.
3. Gunakan `CopyDBClusterSnapshot` untuk menyalin snapshot ke VPC di wilayah cadangan target Anda.
4. Gunakan `RestoreDBClusterFromSnapshot` untuk membuat klaster DB baru di VPC di wilayah cadangan target Anda menggunakan snapshot salinan. Gunakan pengaturan konfigurasi dan parameter yang Anda salin dari klaster produksi utama Anda.
5. Klaster Neptune baru sekarang ada tetapi tidak mengandung instans apapun. Gunakan `CreateDBInstance`untuk membuat primary/writer instance baru yang memiliki jenis dan ukuran instance yang sama dengan instance penulis klaster produksi Anda. Tidak perlu membuat replika baca tambahan pada saat ini kecuali instance cadangan Anda akan digunakan untuk membaca layanan I/O di wilayah target sebelum failover.

Buat peering VPC antara VPC klaster utama Anda dan VPC cluster cadangan baru Anda

Dengan menyiapkan peering VPC, Anda mengaktifkan VPC klaster utama untuk berkomunikasi dengan VPC klaster backup Anda seolah-olah mereka adalah jaringan pribadi tunggal. Untuk melakukannya, Anda harus melakukan langkah-langkah berikut:

1. Dari VPC klaster produksi Anda, panggil `CreateVpcPeeringConnection` API untuk membangun koneksi peering.
2. Dari VPC klaster backup target Anda, panggil API `AcceptVpcPeeringConnection` untuk menerima koneksi peering.
3. Dari VPC klaster produksi Anda, gunakan API `CreateRoute` untuk menambahkan rute ke tabel rute VPC yang mengarahkan semua lalu lintas ke blok CIDR VPC target sehingga menggunakan daftar awalan peering VPC.
4. Demikian pula, dari kluster cadangan target Anda VPC, gunakan API `CreateRoute` untuk menambahkan rute ke tabel rute VPC yang merutekan lalu lintas ke VPC klaster utama.

Siapkan infrastruktur replikasi aliran Neptunus

Sekarang kedua cluster dikerahkan dan komunikasi jaringan antara kedua wilayah telah dibuat, gunakan template Neptunus-ke-Neptunus CloudFormation untuk menyebarkan fungsi Lambda konsumen aliran Neptunus dengan infrastruktur tambahan yang mendukung replikasi data. Lakukan ini di VPC kluster produksi utama Anda.

Parameter yang perlu Anda sediakan untuk CloudFormation tumpukan ini adalah:

- **`NeptuneStreamEndpoint`** – Titik akhir pengaliran untuk klaster utama, dalam format URL. Sebagai contoh: `https://(cluster name):8182/pg/stream`.
- **`QueryEngine`**Ini harus salah satu`gremlin`,`sparql`, atau`openCypher`.
- **`RouteTableIds`** – Memungkinkan Anda menambahkan rute untuk VPC Endpoint DynamoDB dan VPC Endpoint pemantauan.

  Dua parameter tambahan, yaitu `CreateMonitoringEndpoint` dan `CreateDynamoDBEndpoint`, juga harus diatur ke benar jika mereka tidak sudah ada pada VPC klaster utama. Jika mereka sudah ada, pastikan mereka disetel ke false atau CloudFormation kreasi akan gagal.

- **`SecurityGroupIds`** – Menentukan grup keamanan yang digunakan oleh konsumen Lambda untuk berkomunikasi dengan titik akhir pengaliran Neptune klaster utama.

  Di kluster backup target, lampirkan grup keamanan yang memungkinkan lalu lintas yang berasal dari grup keamanan ini.
- **`SubnetIds`** – Daftar ID subnet di VPC klaster utama yang dapat digunakan oleh konsumen Lambda untuk berkomunikasi dengan klaster utama.
- **`TargetNeptuneClusterEndpoint`**— Titik akhir cluster (hanya nama host) dari cluster cadangan target.
- **`TargetAWSRegion`**— AWS Wilayah cluster cadangan target, sepertius`-east-1`). Anda harus memberikan parameter ini hanya ketika AWS wilayah cluster cadangan target berbeda dari wilayah cluster sumber Neptunus, seperti dalam kasus replikasi lintas wilayah. Jika wilayah sumber dan target sama, parameter ini opsional.

  Perhatikan bahwa jika `TargetAWSRegion` nilainya bukan AWS wilayah valid yang didukung Neptunus, prosesnya gagal.
- **`VPC`** – ID VPC klaster utama.

Semua parameter lainnya dapat dibiarkan dengan nilai defaultnya.

Setelah CloudFormation template telah digunakan, Neptunus akan mulai mereplikasi setiap perubahan dari cluster utama ke cluster cadangan. Anda dapat memantau replikasi ini di CloudWatch log yang dihasilkan oleh fungsi konsumen Lambda.

Pertimbangan lainnya

- Jika Anda perlu menggunakan otentikasi IAM antara cluster primer dan cadangan, Anda juga dapat mengaturnya saat Anda memanggil template. CloudFormation
- Jika enkripsi saat istirahat diaktifkan di klaster utama Anda, pertimbangkan cara mengelola kunci KMS yang terkait saat menyalin snapshot di wilayah target dan mengaitkan kunci KMS baru di wilayah target.
- Praktik terbaik adalah menggunakan DNS CNAMEs di depan titik akhir Neptunus yang digunakan dalam aplikasi Anda. Kemudian, jika Anda perlu melakukan failover secara manual ke cluster cadangan target, ini CNAMEs dapat diubah untuk menunjuk ke titik akhir and/or instance cluster target.

Pencarian teks lengkap di Amazon Neptunus menggunakan Layanan Amazon OpenSearch

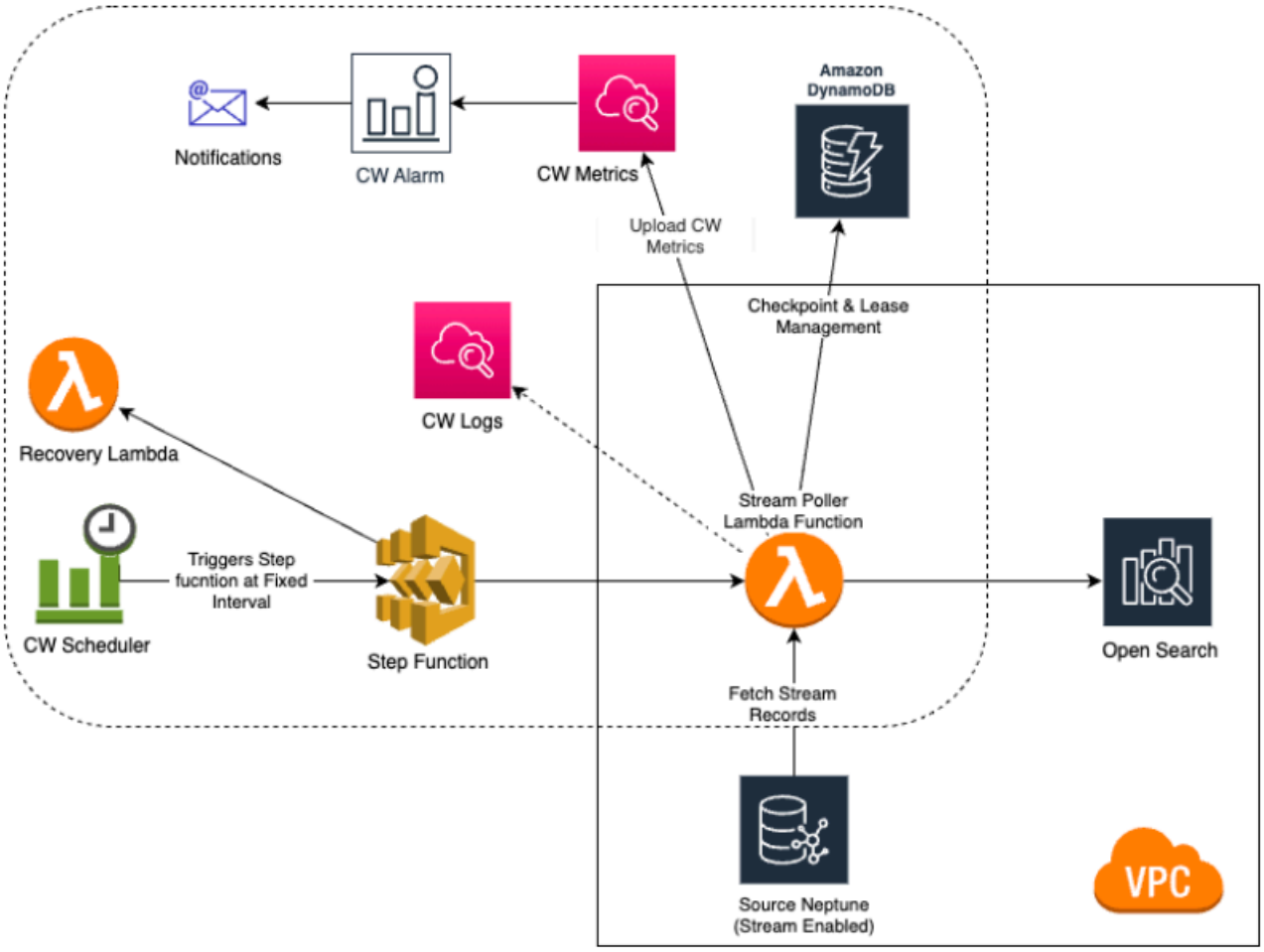
Neptunus terintegrasi dengan OpenSearch Amazon Service OpenSearch (Layanan) untuk mendukung pencarian teks lengkap di kueri Gremlin dan SPARQL.

Dimulai dengan rilis mesin 1.3.0.0, Amazon Neptunus mendukung penggunaan Amazon OpenSearch Service Serverless untuk pencarian teks lengkap dalam kueri Gremlin dan SPARQL.

> **Note**
>
> Saat mengintegrasikan dengan Amazon OpenSearch Service, Neptunus memerlukan Elasticsearch versi 7.1 atau lebih tinggi, dan bekerja dengan 2.3, 2.5 dan lebih tinggi. OpenSearch Neptunus juga bekerja dengan Tanpa Server. OpenSearch

Anda dapat menggunakan Neptunus dengan kluster Layanan yang OpenSearch ada yang telah diisi sesuai dengan. Model data Neptunus untuk data OpenSearch Atau, Anda dapat membuat domain OpenSearch Layanan yang ditautkan dengan Neptunus menggunakan tumpukan. AWS CloudFormation

> **Important**
>
> Proses Neptunus OpenSearch ke replikasi yang dijelaskan di sini tidak mereplikasi node kosong. Ini adalah batasan penting untuk diperhatikan.
> Selain itu, jika Anda mengaktifkan kontrol akses berbutir halus di OpenSearch cluster Anda, Anda perlu mengaktifkan otentikasi IAM di database Neptunus Anda juga.



Topik

- Neptune-to-OpenSearchReplikasi Amazon
- Replikasi ke Tanpa Server OpenSearch
- Kueri dari OpenSearch klaster dengan kontrol akses berbutir halus (FGAC) diaktifkan
- Menggunakan sintaks kueri Apache Lucene di kueri pencarian teks lengkap Neptunus
- Model data Neptunus untuk data OpenSearch
- Parameter pencarian teks lengkap Neptunus
- OpenSearch Pengindeksan non-string di Amazon Neptunus
- Full-text-search eksekusi kueri di Amazon Neptunus
- Contoh kueri SPARQL menggunakan pencarian teks lengkap di Neptunus
- Menggunakan pencarian teks lengkap Neptunus dalam kueri Gremlin
- Memecahkan masalah pencarian teks lengkap Neptunus

Neptune-to-OpenSearchReplikasi Amazon

Amazon Neptunus mendukung pencarian teks lengkap dalam kueri Gremlin dan SPARQL menggunakan Layanan Amazon (Layanan). OpenSearch OpenSearch Anda dapat menggunakan AWS CloudFormation tumpukan untuk menautkan domain OpenSearch Layanan ke Neptunus. CloudFormation Template membuat instance aplikasi streams-consumer yang menyediakan Neptune-to-OpenSearch replikasi.

Sebelum memulai, Anda memerlukan kluster DB Neptunus yang sudah ada dengan aliran yang diaktifkan untuk berfungsi sebagai sumber, dan domain Layanan untuk berfungsi OpenSearch sebagai target replikasi.

Jika Anda sudah memiliki domain OpenSearch Layanan target yang ada yang dapat diakses oleh Lambda di VPC tempat cluster DB Neptunus Anda berada, template dapat menggunakan yang itu. Jika tidak, Anda perlu membuat domain yang baru.

> **Note**
>
> Fungsi OpenSearch cluster dan Lambda yang Anda buat harus berada di VPC yang sama dengan cluster DB Neptunus Anda, dan cluster OpenSearch harus dikonfigurasi dalam mode VPC (bukan mode Internet).

Kami menyarankan Anda menggunakan instance Neptunus yang baru dibuat untuk digunakan dengan Layanan. OpenSearch Jika Anda menggunakan instance yang sudah ada yang sudah memiliki data di dalamnya, Anda harus melakukan sinkronisasi data OpenSearch Layanan sebelum membuat kueri atau mungkin ada inkonsistensi data. GitHub Proyek ini memberikan contoh bagaimana melakukan sinkronisasi: Ekspor Neptunus OpenSearch ke https://github.com/awslabs/amazon-neptune-tools (/-). tree/master/export neptune-to-elasticsearch

> **Important**
>
> Saat berintegrasi dengan Amazon OpenSearch Service, Neptune memerlukan Elasticsearch versi 7.1 atau lebih tinggi, dan bekerja dengan versi Opensearch yang kompatibel dengan OpenSearch 2.3, 2.5 dan future.

> **Note**
>
> Dimulai dengan rilis mesin 1.3.0.0, Amazon Neptunus mendukung penggunaan Amazon OpenSearch Service Serverless untuk pencarian teks lengkap dalam kueri Gremlin dan SPARQL.

Topik

- Menggunakan AWS CloudFormation template untuk memulai Neptune-to-OpenSearch replikasi
- Mengaktifkan pencarian teks lengkap pada database Neptunus yang ada
- Memperbarui poller aliran
- Menonaktifkan dan mengaktifkan kembali proses poller aliran

Menggunakan AWS CloudFormation template untuk memulai Neptune-to-OpenSearch replikasi

Luncurkan CloudFormation tumpukan khusus untuk wilayah Anda

Masing-masing CloudFormation template di bawah ini membuat instance aplikasi streams-consumer di wilayah tertentu AWS . Untuk meluncurkan tumpukan yang sesuai menggunakan CloudFormation konsol, pilih salah satu tombol Launch Stack di tabel berikut, tergantung pada AWS Wilayah yang ingin Anda gunakan.

Region	Lihat	Lihat di Designer	Luncurkan
US East (N. Virginia)	Lihat	Lihat di Desainer	Launch Stack
AS Timur (Ohio)	Lihat	Lihat di Desainer	Launch Stack
US West (N. Californi a)	Lihat	Lihat di Desainer	Launch Stack
US West (Oregon)	Lihat	Lihat di Desainer	Launch Stack

Region	Lihat	Lihat di Designer	Luncurkan
Kanada (Pusat)	Lihat	Lihat di Desainer	Launch Stack
Amerika Selatan (Sao Paulo)	Lihat	Lihat di Desainer	Launch Stack
Europe (Stockholm)	Lihat	Lihat di Desainer	Launch Stack
Eropa (Irlandia)	Lihat	Lihat di Desainer	Launch Stack
Eropa (London)	Lihat	Lihat di Desainer	Launch Stack
Europe (Paris)	Lihat	Lihat di Desainer	Launch Stack
Eropa (Spanyol)	Lihat	Lihat di Desainer	Launch Stack
Eropa (Frankfurt)	Lihat	Lihat di Desainer	Launch Stack
Timur Tengah (Bahrain)	Lihat	Lihat di Desainer	Launch Stack
Timur Tengah (UAE)	Lihat	Lihat di Desainer	Launch Stack
Israel (Tel Aviv)	Lihat	Lihat di Desainer	Launch Stack
Africa (Cape Town)	Lihat	Lihat di Desainer	Launch Stack
Asia Pasifik (Hong Kong)	Lihat	Lihat di Desainer	Launch Stack
Asia Pacific (Tokyo)	Lihat	Lihat di Desainer	Launch Stack

Region	Lihat	Lihat di Designer	Luncurkan
Asia Pasifik (Seoul)	Lihat	Lihat di Desainer	Launch Stack
Asia Pacific (Singapore)	Lihat	Lihat di Desainer	Launch Stack
Asia Pasifik (Jakarta)	Lihat	Lihat di Desainer	Launch Stack
Asia Pasifik (Mumbai)	Lihat	Lihat di Desainer	Launch Stack
Tiongkok (Beijing)	Lihat	Lihat di Desainer	Launch Stack
Tiongkok (Ningxia)	Lihat	Lihat di Desainer	Launch Stack
AWS GovCloud (AS-Barat)	Lihat	Lihat di Desainer	Launch Stack
AWS GovCloud (AS-Timur)	Lihat	Lihat di Desainer	Launch Stack

Pada halaman Buat Tumpukan, pilih Selanjutnya.

Tambahkan Detail Tentang OpenSearch tumpukan baru yang Anda buat

Halaman Tentukan Detail Tumpukan menyediakan properti dan parameter yang dapat Anda gunakan untuk mengontrol pengaturan pencarian teks lengkap:

Nama Stack — Nama CloudFormation tumpukan baru yang Anda buat. Anda dapat menggunakan nilai default secara umum, `NeptuneStreamPoller`.

Di bawah Parameter, berikan yang berikut ini:

Konfigurasi jaringan untuk VPC tempat Konsumen Stream berjalan

- **VPC**— Berikan nama VPC tempat fungsi Lambda polling akan berjalan.

- **`List of Subnet IDs`**— Subnet tempat jaringan antarmuka didirikan. Tambahkan subnet yang sesuai dengan klaster Neptune Anda.
- **`List of Security Group Ids`**— Berikan grup keamanan IDs yang memberikan akses masuk tulis ke cluster DB Neptunus sumber Anda.
- **`List of Route Table Ids`**— Ini diperlukan untuk membuat titik akhir Amazon DynamoDB di VPC Neptune Anda, jika Anda belum memilikinya. Anda harus memberikan daftar tabel rute yang dipisahkan koma yang IDs terkait dengan subnet.
- **`Require to create Dynamo DB VPC Endpoint`**— Nilai Boolean yang default ke `true`. Anda hanya perlu mengubahnya menjadi `false` jika Anda telah membuat titik akhir DynamoDB di VPC Anda.
- **`Require to create Monitoring VPC Endpoint`**— Nilai Boolean yang default ke `true`. Anda hanya perlu mengubahnya menjadi `false` jika Anda telah membuat titik akhir pemantauan di VPC Anda.

Poller Stream

- **`Application Name`**— Anda umumnya dapat meninggalkan aturan ini ke default (`NeptuneStream`). Jika Anda menggunakan nama yang berbeda, itu harus unik.
- **`Memory size for Lambda Poller`**— Digunakan untuk mengatur ukuran memori yang tersedia untuk fungsi poller Lambda. Nilai default-nya adalah 2,048 megabyte.
- **`Lambda Runtime`**— Bahasa yang digunakan dalam fungsi Lambda yang mengambil item dari aliran Neptunus. Anda dapat mengatur ini baik ke `python3.9` atau ke `java8`.
- **`S3 Bucket having Lambda code artifacts`**— Tinggalkan ini kosong kecuali Anda menggunakan fungsi polling Lambda kustom yang memuat dari bucket S3 yang berbeda.
- **`S3 Key corresponding to Lambda Code artifacts`**— Tinggalkan ini kosong kecuali Anda menggunakan fungsi polling Lambda kustom.
- **`StartingCheckpoint`**— pos pemeriksaan awal untuk poller aliran. Defaultnya adalah`0:0`, yang menandakan mulai dari awal aliran Neptunus.
- **`StreamPollerInitialState`**- Keadaan awal poller. Defaultnya adalah`ENABLED`, yang berarti replikasi aliran akan dimulai segera setelah seluruh pembuatan tumpukan selesai.
- **`Logging level for Lambda`**— Umumnya, tinggalkan aturan ini ke nilai default, `INFO`.
- **`Managed Policies for Lambda Execution`**— Umumnya, tinggalkan ini kosong kecuali Anda menggunakan fungsi polling Lambda kustom.

- **`Stream Records Handler`**— Secara umum, biarkan kosong ini kecuali Anda menggunakan penangan khusus untuk catatan di aliran Neptunus.
- **`Maximum records Fetched from Stream`**— Anda dapat menggunakan parameter ini untuk menyetel performa. Default (`100`) adalah tempat yang baik untuk memulai. Maksimum yang diijinkan adalah 10.000. Semakin tinggi jumlahnya, semakin sedikit panggilan jaringan yang diperlukan untuk membaca catatan dari stream, tetapi semakin banyak memori diperlukan untuk memproses catatan.
- **`Max wait time between two Polls (in Seconds)`**— Menentukan seberapa sering poller Lambda dipanggil untuk polling stream Neptune. Tetapkan nilai ini ke 0 untuk polling berkelanjutan. Nilai maksimum adalah 3.600 detik (1 jam). Nilai default (60 detik) adalah tempat yang baik untuk memulai, tergantung seberapa cepat data grafik Anda berubah.
- **`Maximum Continuous polling period (in Seconds)`**- Digunakan untuk mengatur batas waktu untuk fungsi polling Lambda. Seharusnya antara 5 detik dan 900 detik. Nilai default (600 detik) adalah tempat yang baik untuk memulai.
- **`Step Function Fallback Period`**— Jumlah step-function-fallback-period unit yang menunggu poller, setelah itu fungsi langkah dipanggil melalui Amazon CloudWatch Events untuk pulih dari kegagalan. Default (5 menit) adalah tempat yang baik untuk memulai.
- **`Step Function Fallback Period Unit`**— Satuan waktu yang digunakan untuk mengukur sebelumnya `Step Function Fallback Period` (menit, jam, hari). Default (menit) umumnya cukup.
- **`Enable/disable creation of Elastic Search Mappings by Lambda`**— Tandai ke manajemen enable/disable otomatis pemetaan Pencarian Elastis oleh poller aliran. Pemetaan Elastic Search sangat penting untuk konversi antara tipe data Neptunus dan Elastic Search. Ini hanya boleh dinonaktifkan oleh pengguna tingkat lanjut yang ingin mengambil kontrol manual langsung atas semua pemetaan dalam indeks Pencarian Elastis mereka. Nilai default-nya `true`.
- **`Data replication scope`**— Menentukan apakah akan mereplikasi kedua node dan tepi, atau hanya node ke OpenSearch (ini hanya berlaku untuk data mesin Gremlin). Nilai default (Semua) umumnya tempat yang baik untuk memulai.
- **`Ignore OpenSearch missing document error`**— Tandai untuk menentukan apakah kesalahan dokumen yang hilang OpenSearch dapat diabaikan. Kesalahan dokumen yang hilang jarang terjadi tetapi perlu intervensi manual jika tidak diabaikan. Nilai default (`True`) umumnya merupakan tempat yang baik untuk memulai.
- **`Enable Non-String Indexing`**- Tandai untuk mengaktifkan atau menonaktifkan pengindeksan bidang yang tidak memiliki konten string. Jika bendera ini disetel ke`true`, bidang non-string diindeks OpenSearch, atau jika`false`, hanya bidang string yang diindeks. Nilai default-nya `true`.

- **`Properties to exclude from being inserted into OpenSearch`**— Daftar properti atau kunci predikat yang dibatasi koma untuk dikecualikan dari pengindeksan. OpenSearch Jika nilai parameter CFN ini dibiarkan kosong, semua kunci properti diindeks.
- **`Datatypes to exclude from being inserted into OpenSearch`**— Daftar properti atau tipe data predikat yang dibatasi koma untuk dikecualikan dari pengindeksan. OpenSearch Jika nilai parameter CFN ini dibiarkan kosong, semua nilai properti yang dapat dikonversi dengan aman ke OpenSearch tipe data diindeks.

Stream Neptune

- **`Endpoint of source Neptune Stream`**— (Wajib) Ini mengambil salah satu dari dua bentuk:
  - **`https://`*`your DB cluster`*`:`*`port`*`/propertygraph/stream`**(atau aliasnya,`https://`*`your DB cluster`*`:`*`port`*`/pg/stream`).
  - `https://`*`your DB cluster`*`:`*`port`*`/sparql/stream`
- **`Neptune Query Engine`**— Pilih Gremlin atau SPARQL.
- **`Is IAM Auth Enabled?`**— Jika kluster DB Neptune Anda menggunakan otentikasi IAM, atur parameter ini ke `true`.
- **`Neptune Cluster Resource Id`** – Jika kluster DB Neptune Anda menggunakan otentikasi IAM, atur parameter ini ke ID sumber daya. ID sumber daya tidak sama dengan ID cluster. Sebaliknya, ia mengambil bentuk: `cluster-` diikuti oleh 28 karakter alfa-numerik. Ini dapat ditemukan di bawah Detail Klaster di konsol Neptune.

OpenSearch Cluster target

- **`Endpoint for OpenSearch service`**— (Wajib) Berikan titik akhir untuk OpenSearch layanan di VPC Anda.
- **`Number of Shards for OpenSearch Index`** – Nilai default (5) umumnya tempat yang baik untuk memulai.
- **`Number of Replicas for OpenSearch Index`** – Nilai default (1) umumnya tempat yang baik untuk memulai.
- **`Geo Location Fields for Mapping`**— Jika Anda menggunakan bidang geolokasi, daftar kunci properti di sini.

Alarm

- **`Require to create Cloud watch Alarm`**— Atur ini ke `true` jika Anda ingin membuat CloudWatch alarm untuk tumpukan baru.
- **`SNS Topic ARN for Cloudwatch Alarm Notifications`**— Topik SNS ARN CloudWatch tempat pemberitahuan alarm harus dikirim (hanya diperlukan jika alarm diaktifkan).
- **`Email for Alarm Notifications`**— Alamat email yang harus dikirim notifikasi alarm (hanya diperlukan jika alarm diaktifkan).

Untuk tujuan notifikasi alarm, Anda dapat menambahkan SNS saja, hanya email, atau SNS dan email.

Jalankan CloudFormation Template

Sekarang Anda dapat menyelesaikan proses instans aplikasi konsumen stream Neptune provisioning sebagai berikut:

1. Di CloudFormation, pada halaman Tentukan Detail Tumpukan, pilih Berikutnya.
2. Pada halaman Opsi, pilih Selanjutnya.
3. Pada halaman Review, pilih kotak centang pertama untuk mengetahui bahwa CloudFormation akan membuat sumber daya IAM. Pilih kotak centang kedua untuk mengetahui `CAPABILITY_AUTO_EXPAND` untuk tumpukan baru.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` secara eksplisit mengakui bahwa macro akan diperluas saat membuat tumpukan, tanpa review sebelumnya. Pengguna sering kali membuat perubahan yang ditetapkan dari templat yang diproses, sehingga perubahan yang dibuat oleh makro bisa direview tepat sebelum membuat tumpukan. Untuk informasi selengkapnya, lihat operasi CloudFormation CreateStackAPI di Referensi AWS CloudFormation API.

   Lalu pilih Buat.

Mengaktifkan pencarian teks lengkap pada database Neptunus yang ada

Ini adalah pendekatan yang ditetapkan untuk memungkinkan pencarian teks lengkap pada database Amazon Neptunus yang ada. Bergantung pada apakah Anda dapat menjeda beban kerja tulis Anda atau tidak, langkah-langkahnya mungkin sedikit berbeda. Panduan ini menguraikan langkah-langkah yang disarankan untuk kedua skenario - kapan Anda dapat menjeda penulisan, dan kapan Anda tidak bisa. Ini mencakup mengaktifkan aliran Neptunus, membuat klon database, menyinkronkan data ke domain, dan menyiapkan OpenSearch pembaruan berkelanjutan. Panduan ini memanfaatkan AWS layanan dan alat sumber terbuka untuk merampingkan proses dan meminimalkan waktu henti.

Jika Anda dapat menjeda beban kerja tulis

Cara terbaik untuk mengaktifkan pencarian teks lengkap pada database Neptunus yang ada umumnya sebagai berikut, asalkan Anda dapat menjeda beban kerja tulis Anda. Ini membutuhkan pembuatan klon, mengaktifkan aliran menggunakan parameter cluster, dan memulai ulang semua instance. Membuat klon adalah operasi yang relatif cepat, sehingga waktu henti yang diperlukan terbatas.

Berikut langkah-langkah yang diperlukan:

1. Hentikan semua beban kerja tulis pada database.
2. Aktifkan aliran pada database (lihat Mengaktifkan Aliran Neptunus).
3. Buat klon database (lihat Kloning Database di Neptunus).
4. Lanjutkan beban kerja tulis.
5. Gunakan export-neptune-to-elasticsearchalat di github untuk melakukan sinkronisasi satu kali dari database kloning ke domain. OpenSearch
6. Gunakan CloudFormation template untuk wilayah Anda untuk memulai sinkronisasi dari database asli Anda dengan pembaruan berkelanjutan (tidak diperlukan perubahan konfigurasi dalam template).
7. Hapus database kloning dan CloudFormation tumpukan yang dibuat untuk `export-neptune-to-elasticsearch` alat.

> **Note**
>
> Jika menggunakan export-neptune-to-elasticsearchOpensearch Tanpa Server, yang `LambdaExecutionRole` dibuat oleh tumpukan export-neptune-to-

elasticsearchCloudformation harus ditambahkan ke Kebijakan Akses Data Opensearch agar berhasil mereplikasi data.

Jika Anda tidak dapat menjeda beban kerja tulis Anda

Jika Anda tidak mampu menangguhkan beban kerja tulis pada database Anda, berikut adalah pendekatan yang membutuhkan waktu henti lebih sedikit daripada pendekatan yang disarankan di atas, tetapi perlu dilakukan dengan hati-hati:

1. Aktifkan aliran pada database (lihat Mengaktifkan Aliran Neptunus).
2. Buat klon database (lihat Kloning Database di Neptunus).
3. Dapatkan yang terbaru `eventID` untuk aliran pada database kloning dengan menjalankan perintah semacam ini terhadap titik akhir Streams API (lihat Memanggil REST API Neptunus Streams untuk informasi selengkapnya):

   ```
   curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
   iteratorType=LATEST"
   ```

   Catat nilai-nilai di `commitNum` dan opNum bidang di `lastEventId` objek dalam respons.
4. Gunakan export-neptune-to-elasticsearchalat di github untuk melakukan sinkronisasi satu kali dari database kloning ke domain. OpenSearch
5. Gunakan CloudFormation template untuk wilayah Anda untuk memulai sinkronisasi dari database asli Anda dengan pembaruan berkelanjutan.

   Buat perubahan berikut saat membuat tumpukan: pada halaman detail tumpukan, di bagian Parameter, atur nilai `StartingCheckpoint` bidang menjadi*`commitNum`*: *`opnum`* menggunakan opNum nilai `commitNum` dan yang Anda rekam di atas.
6. Hapus database kloning dan CloudFormation tumpukan yang dibuat untuk `export-neptune-to-elasticsearch` alat.

Memperbarui poller aliran

Informasi berikut menguraikan langkah-langkah yang diperlukan untuk memperbarui poller aliran dengan artefak Lambda terbaru menggunakan konsol manajemen. AWS

Untuk memperbarui poller aliran dengan artefak Lambda terbaru

Anda dapat memperbarui poller aliran dengan artefak kode Lambda terbaru sebagai berikut:

1. Di Konsol Manajemen AWS, navigasikan ke CloudFormation dan pilih CloudFormation tumpukan induk utama.
2. Pilih opsi Perbarui untuk tumpukan.
3. Pilih Ganti template saat ini.
4. Untuk sumber template, pilih URL Amazon S3 dan masukkan URL S3 berikut:

```
https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
neptune_to_elastic_search.json
```

5. Pilih Berikutnya tanpa mengubah CloudFormation parameter apa pun.
6. Pilih Update Stack.

Tumpukan sekarang akan memperbarui artefak Lambda dengan yang terbaru.

Memperluas stream poller untuk mendukung bidang kustom

Poller aliran saat ini dapat dengan mudah diperluas untuk menulis kode khusus untuk menangani bidang khusus, seperti yang dijelaskan secara rinci dalam posting blog ini: Tangkap perubahan grafik menggunakan Aliran Neptunus.

> **Note**
>
> Saat menambahkan bidang khusus OpenSearch, pastikan untuk menambahkan bidang baru sebagai objek dalam predikat (lihatModel data pencarian teks lengkap Neptunus).

Menonaktifkan dan mengaktifkan kembali proses poller aliran

> **Warning**
>
> Hati-hati saat Anda menonaktifkan proses poller aliran! Kehilangan data dapat terjadi jika proses dijeda lebih lama dari jendela kedaluwarsa aliran. Jendela default adalah 7 hari, tetapi

dimulai dengan versi mesin 1.2.0.0, Anda dapat mengatur jendela kedaluwarsa aliran kustom hingga maksimum 90 hari.

Menonaktifkan (menjeda) proses poller aliran

1. Masuk ke Konsol Manajemen AWS dan buka EventBridge konsol Amazon di https://console.aws.amazon.com/events/.
2. Di panel navigasi, pilih Aturan.
3. Pilih aturan yang namanya berisi nama yang Anda berikan sebagai Nama Aplikasi dalam CloudFormation template yang Anda gunakan untuk mengatur poller aliran.
4. Pilih Disable (Nonaktifkan).
5. Buka konsol Step Functions di https://console.aws.amazon.com/states/.
6. Pilih fungsi langkah berjalan yang sesuai dengan proses poller aliran. Sekali lagi, nama fungsi langkah itu berisi nama yang Anda berikan sebagai Nama Aplikasi dalam CloudFormation template yang Anda gunakan untuk mengatur poller aliran. Anda dapat memfilter berdasarkan status eksekusi fungsi untuk melihat hanya fungsi Menjalankan.
7. Pilih Berhenti.

Mengaktifkan kembali proses poller aliran

1. Masuk ke Konsol Manajemen AWS dan buka EventBridge konsol Amazon di https://console.aws.amazon.com/events/.
2. Di panel navigasi, pilih Aturan.
3. Pilih aturan yang namanya berisi nama yang Anda berikan sebagai Nama Aplikasi dalam CloudFormation template yang Anda gunakan untuk mengatur poller aliran.
4. Pilih Aktifkan. Aturan acara berdasarkan interval terjadwal yang ditentukan sekarang akan memicu eksekusi baru dari fungsi langkah.

Replikasi ke Tanpa Server OpenSearch

Dimulai dengan rilis mesin 1.3.0.0, Amazon Neptunus mendukung penggunaan Amazon OpenSearch Service Serverless untuk pencarian teks lengkap dalam kueri Gremlin dan SPARQL. Menggunakan OpenSearch Tanpa Server mengharuskan Anda untuk mengaktifkan otentikasi IAM di cluster

Database Neptunus Anda. Cluster Neptunus Database dengan autentikasi IAM dinonaktifkan tidak didukung dengan Tanpa Server. OpenSearch

Perubahan yang diperlukan untuk fungsi AWS Lambda poller

Jika Anda mereplikasi ke OpenSearch Tanpa Server, tambahkan peran eksekusi poller aliran Lambda ke kebijakan akses data untuk koleksi Tanpa Server. OpenSearch ARN untuk peran eksekusi poller aliran Lambda memiliki format ini:

```
arn:aws:iam::(account ID):role/stack-name-NeptuneOSReplication-
NeptuneStreamPollerExecu-(uuid)
```

Perubahan yang diperlukan untuk utilitas impor massal

Jika Anda menggunakan [export-neptune-to-elasticsearch](#)untuk menyinkronkan data yang ada ke OpenSearch Tanpa Server, tambahkan `LambdaExecutionRole` dari CloudFormation tumpukan ke kebijakan akses data untuk koleksi Tanpa Server. OpenSearch ARN untuk `LambdaExecutionRole` memiliki format ini:

```
arn:aws:iam::012345678901:role/stack-name-LambdaExecutionRole-(id)
```

Untuk informasi selengkapnya, lihat [Kontrol akses data untuk Amazon Tanpa OpenSearch Server](#).

Perubahan yang diperlukan pada peran IAM yang digunakan untuk menanyakan Neptunus

Entitas IAM (Pengguna atau Peran) yang digunakan untuk menghubungkan ke database Neptunus harus memiliki izin baik untuk Neptunus dan koleksi Tanpa Server. OpenSearch Ini berarti bahwa pengguna atau peran Anda harus memiliki kebijakan OpenSearch Tanpa Server seperti ini terlampir:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Sid": "AllowOpenSearchServerlessAccess",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "aoss:APIAccessAll",
      "Resource": "arn:aws:aoss:us-east-1:111122223333:collection/collection-id"
    }
  ]
}
```

Lihat Membuat pernyataan kebijakan IAM khusus untuk mengakses data di Amazon Neptunus untuk informasi selengkapnya.

Kueri dari OpenSearch klaster dengan kontrol akses berbutir halus (FGAC) diaktifkan

Jika Anda telah mengaktifkan kontrol akses berbutir halus di OpenSearch cluster Anda, Anda perlu mengaktifkan otentikasi IAM di database Neptunus Anda juga.

Entitas IAM (Pengguna atau Peran) yang digunakan untuk menghubungkan ke database Neptunus harus memiliki izin baik untuk Neptunus dan cluster. OpenSearch Ini berarti bahwa pengguna atau peran Anda harus memiliki kebijakan OpenSearch Layanan seperti ini terlampir:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFullTextSearchAccess",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "es:*",
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/domain-name/*"
```

```
    }
  ]
}
```

Lihat Membuat pernyataan kebijakan IAM khusus untuk mengakses data di Amazon Neptunus untuk informasi selengkapnya.

Menggunakan sintaks kueri Apache Lucene di kueri pencarian teks lengkap Neptunus

OpenSearch mendukung penggunaan sintaks Apache Lucene untuk kueri query_string. Ini sangat berguna untuk meneruskan beberapa filter dalam kueri.

Neptunus menggunakan struktur bersarang untuk menyimpan properti dalam OpenSearch dokumen (lihat). Model data pencarian teks lengkap Neptunus Saat menggunakan sintaks Lucene, Anda perlu menggunakan jalur lengkap ke properti dalam model berikutnya ini.

Berikut adalah contoh Gremlin:

```
g.withSideEffect("Neptune#fts.endpoint", "es_endpoint")
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V()
 .has("*", "Neptune#fts predicates.name.value:\"Jane Austin\" AND entity_type:Book")
```

Berikut adalah contoh SPARQL:

```
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
  neptune-fts:config neptune-fts:endpoint 'http://localhost:9200 (http://
localhost:9200/)' .
  neptune-fts:config neptune-fts:queryType 'query_string' .
  neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*name.value:Ronak AND
 predicates.\\*foaf\\*surname.value:Sh*" .
  neptune-fts:config neptune-fts:field '*' .
  neptune-fts:config neptune-fts:return ?res .
}
```

Model data Neptunus untuk data OpenSearch

Amazon Neptunus menggunakan struktur dokumen JSON terpadu untuk menyimpan data SPARQL dan Gremlin dalam Layanan. OpenSearch Setiap dokumen OpenSearch sesuai dengan entitas dan menyimpan semua informasi yang relevan untuk entitas tersebut. Untuk Gremlin, simpul dan tepi dianggap entitas, sehingga OpenSearch dokumen yang sesuai memiliki informasi tentang simpul, label, dan properti. Untuk SPARQL, subjek dapat dianggap entitas, sehingga OpenSearch dokumen yang sesuai memiliki informasi tentang semua pasangan objek predikat dalam satu dokumen.

> **Note**
>
> Neptune-to-OpenSearchImplementasi replikasi hanya menyimpan data string. Namun, Anda dapat mengubahnya untuk menyimpan jenis data lain.

Struktur dokumen JSON terpadu terlihat seperti berikut ini.

```
{
  "entity_id": "Vertex Id/Edge Id/Subject URI",
  "entity_type": [List of Labels/rdf:type object value],
  "document_type": "vertex/edge/rdf-resource"
  "predicates": {
    "Property name or predicate URI": [
      {
        "value": "Property Value or Object Value",
        "graph": "(Only for Sparql) Named Graph Quad is present"
        "language": "(Only for Sparql) rdf:langString"
      },
      {
        "value": "Property Value 2/ Object Value 2",
      }
    ]
  }
}
```

- `entity_id` — ID unik entitas yang mewakili dokumen.
  - Untuk SPARQL, ini adalah URI subjek.
  - Untuk Gremlin, ini adalah `Vertex_ID` atau `Edge_ID`.

- `entity_type` — Mewakili satu label atau lebih untuk vertex atau edge, atau nilai predikat `rdf:type` nol atau lebih untuk subjek.
- `document_type` — Digunakan untuk menentukan apakah dokumen saat ini merupakan vertex, edge, atau rdf-sumber daya.
- `predicates` — Untuk Gremlin, menyimpan properti dan nilai-nilai untuk vertex atau edge. Untuk SPARQL, menyimpan pasangan predikat-objek.

  Nama properti mengambil formulir `properties.name.value` di OpenSearch. Untuk mengkuerinya, Anda harus memberi namanya dalam bentuk itu.
- `value` — Sebuah nilai properti untuk Gremlin atau nilai objek untuk SPARQL.
- `graph` — Sebuah grafik bernama untuk SPARQL.
- `language` — Tanda bahasa untuk literal `rdf:langString` dalam SPARQL.

Contoh dokumen OpenSearch SPARQL

Data

```
@prefix dt:   <http://example.org/datatype#> .
@prefix ex:   <http://example.org/> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

ex:simone   rdf:type     ex:Person                   ex:g1
ex:michael  rdf:type     ex:Person                   ex:g1
ex:simone   ex:likes     "spaghetti"                 ex:g1

ex:simone   ex:knows     ex:michael                  ex:g2   # Not stored in ES
ex:simone   ex:likes     "spaghetti"                 ex:g2
ex:simone   ex:status    "La vita è un sogno"@it     ex:g2

ex:simone   ex:age       "40"^^xsd:int               DG      # Not stored in ES
ex:simone   ex:dummy     "testData"^^dt:newDataType  DG      # Not stored in ES
ex:simone   ex:hates     _:bnode                             # Not stored in ES
_:bnode     ex:means     "coding"                    DG      # Not stored in ES
```

Dokumen

```
{
  "entity_id": "http://example.org/simone",
```

```
  "entity_type": ["http://example.org/Person"],
  "document_type": "rdf-resource"
  "predicates": {
    "http://example.org/likes": [
      {
        "value": "spaghetti",
        "graph": "http://example.org/g1"
      },
      {
        "value": "spaghetti",
        "graph": "http://example.org/g2"
      }
    ]
    "http://example.org/status": [
      {
        "value": "La vita è un sogno",
        "language": "it"       // Only present for rdf:langString
      }
    ]
  }
}
```

```
{
  "entity_id" : "http://example.org/michael",
  "entity_type" : ["http://example.org/Person"],
  "document_type": "rdf-resource"
}
```

Contoh dokumen Gremlin OpenSearch

Data

```
# Vertex 1
simone   label    Person       <== Label
simone   likes    "spaghetti"  <== Property
simone   likes    "rice"       <== Property
simone   age      40           <== Property

# Vertex 2
michael  label    Person       <== Label

# Edge 1
simone  knows     michael      <== Edge
```

```
e1      updated  "2019-07-03"  <== Edge Property
e1      through  "company"     <== Edge Property
e1      since     10           <== Edge Property
```

Dokumen

```
{
  "entity_id": "simone",
  "entity_type": ["Person"],
  "document_type": "vertex",
  "predicates": {
    "likes": [
      {
        "value": "spaghetti"
      },
      {
        "value": "rice"
      }
    ]
  }
}
```

```
{
  "entity_id" : "michael",
  "entity_type" : ["Person"],
  "document_type": "vertex"
}
```

```
{
  "entity_id": "e1",
  "entity_type": ["knows"],
  "document_type": "edge"
  "predicates": {
    "through": [
      {
        "value": "company"
      }
    ]
  }
}
```

Parameter pencarian teks lengkap Neptunus

Amazon Neptunus menggunakan parameter berikut untuk menentukan kueri OpenSearch teks lengkap di Gremlin dan SPARQL:

- **`queryType`**— (Diperlukan) Jenis OpenSearch kueri. (Untuk daftar jenis kueri, lihat OpenSearch dokumentasi). Neptunus mendukung jenis kueri berikut OpenSearch :
  - simple_query_string — Mengembalikan dokumen berdasarkan string kueri yang disediakan, menggunakan pengurai dengan sintaks Lucene terbatas tapi toleran kesalahan. Ini adalah jenis kueri default.

    Kueri ini menggunakan sintaks sederhana untuk mengurai dan membagi string kueri yang disediakan menjadi istilah berdasarkan operator khusus. Kueri kemudian menganalisis setiap istilah secara independen sebelum mengembalikan dokumen yang cocok.

    Sementara sintaksnya lebih terbatas dari kueri `query_string`, kueri `simple_query_string` tidak mengembalikan kesalahan untuk sintaks yang tidak valid. Sebaliknya, kueri tersebut mengabaikan setiap bagian yang tidak valid dari string kueri.
  - match — Kueri `match` adalah kueri standar untuk melakukan pencarian teks lengkap, termasuk opsi untuk pencocokan fuzzy.
  - prefix — Mengembalikan dokumen yang berisi prefiks tertentu di bidang yang disediakan.
  - fuzzy — Mengembalikan dokumen yang berisi istilah yang mirip dengan istilah pencarian, yang diukur dengan jarak edit Levenshtein.

    Jarak edit adalah jumlah perubahan satu karakter yang diperlukan untuk mengubah satu istilah ke istilah lain. Perubahan ini dapat mencakup:
    - Mengubah karakter (box ke fox).
    - Menghapus karakter (black ke lack).
    - Memasukkan karakter (sic ke sick).
    - Mengubah urutan dua karakter yang berdekatan (act ke cat).

    Untuk menemukan istilah yang sama, kueri fuzzy menciptakan serangkaian semua kemungkinan variasi dan perluasan istilah pencarian dalam jarak edit yang ditentukan dan kemudian mengembalikan kecocokan tepat untuk masing-masing varian tersebut.
  - term — Mengembalikan dokumen yang berisi kecocokan tepat dari istilah tertentu di salah satu bidang yang ditentukan.

Anda dapat menggunakan kueri `term` untuk menemukan dokumen berdasarkan nilai yang tepat seperti harga, ID produk, atau nama pengguna.

> ⚠ Warning
>
> Hindari menggunakan kueri istilah untuk bidang teks. Secara default, OpenSearch mengubah nilai bidang teks sebagai bagian dari analisisnya, yang dapat membuat sulit menemukan kecocokan yang tepat untuk nilai bidang teks.
> Untuk mencari nilai bidang teks, gunakan kueri match sebagai gantinya.

- simple_query_string — Mengembalikan dokumen berdasarkan string kueri yang disediakan, menggunakan pengurai dengan sintaks ketat (sintaks Lucene).

  Kueri ini menggunakan sintaks untuk mengurai dan membagi string kueri yang disediakan berdasarkan operator, seperti AND dan NOT. Kueri kemudian menganalisis setiap istilah secara independen sebelum mengembalikan dokumen yang cocok.

  Anda dapat menggunakan `query_string` untuk membuat pencarian kompleks yang mencakup karakter wildcard, pencarian di beberapa bidang, dan lainnya. Meski serbaguna, kueri tersebut ketat dan mengembalikan kesalahan jika string kueri menyertakan sintaks yang tidak valid.

  > ⚠ Warning
  >
  > Karena kueri tersebut mengembalikan kesalahan untuk sintaks yang tidak valid, kami tidak menyarankan menggunakan kueri `query_string` untuk kotak pencarian.
  > Jika Anda tidak perlu mendukung sintaks kueri, pertimbangkan untuk menggunakan kueri `match`. Jika Anda membutuhkan fitur sintaks kueri, gunakan kueri `simple_query_string`, yang kurang ketat.

- **`field`**— Bidang OpenSearch yang digunakan untuk menjalankan pencarian. Hal ini dapat dihilangkan hanya jika `queryType` mengizinkannya (sebagaimana `simple_query_string` dan `query_string` lakukan), dalam hal ini pencarian terhadap semua bidang. Di Gremlin, ini implisit.

  Beberapa bidang dapat ditentukan jika kueri mengizinkannya, seperti halnya `simple_query_string` dan `query_string`.

- **`query`**— (Wajib) Kueri untuk dijalankan terhadap OpenSearch. Isi dari bidang ini dapat bervariasi sesuai dengan QueryType. QueryTypes berbeda menerima sintaks yang berbeda, sebagaimana `Regexp` lakukan, misalnya. Di Gremlin, `query` implisit.
- **`maxResults`** — Jumlah maksimum hasil yang akan dikembalikan. Defaultnya adalah `index.max_result_window` OpenSearch pengaturan, yang dengan sendirinya default menjadi 10.000. Parameter `maxResults` dapat menentukan angka yang lebih rendah dari itu.

  > **Important**
  >
  > Jika Anda menyetel `maxResults` ke nilai yang lebih tinggi dari OpenSearch `index.max_result_window` nilai dan mencoba mengambil lebih dari `index.max_result_window` hasil, OpenSearch gagal dengan `Result window is too large` kesalahan. Namun, Neptune menangani ini dengan anggun tanpa menyebarkan kesalahan. Ingatlah hal ini jika Anda mencoba untuk mengambil lebih dari `index.max_result_window` hasil.

- **`minScore`** — Nilai minimum hasil pencarian harus dikembalikan. Lihat dokumentasi OpenSearch relevansi untuk penjelasan penilaian hasil.
- **`batchSize`** — Neptune selalu mengambil data dalam batch (ukuran batch default adalah 100). Anda dapat menggunakan parameter ini untuk menyetel performa. Ukuran batch tidak dapat melebihi `index.max_result_window` OpenSearch pengaturan, yang defaultnya 10.000.
- **`sortBy`**— Parameter opsional yang memungkinkan Anda mengurutkan hasil yang dikembalikan OpenSearch oleh salah satu dari berikut ini:
  - Bidang string tertentu dalam dokumen -

    Sebagai contoh, dalam kueri SPARQL, Anda dapat menentukan:

    ```
    neptune-fts:config neptune-fts:sortBy foaf:name .
    ```

    Dalam kueri Gremlin yang sama, Anda bisa menentukan:

    ```
    .withSideEffect('Neptune#fts.sortBy', 'name')
    ```

  - Bidang non-string tertentu (*`long`*,*`double`*, dll.) dalam dokumen —

    Perhatikan bahwa saat menyortir bidang non-string, Anda perlu menambahkan `.value` ke nama bidang untuk membedakannya dari bidang string.

Sebagai contoh, dalam kueri SPARQL, Anda dapat menentukan:

```
    neptune-fts:config neptune-fts:sortBy foaf:name.value .
```

Dalam kueri Gremlin yang sama, Anda bisa menentukan:

```
    .withSideEffect('Neptune#fts.sortBy', 'name.value')
```

- `score` — Urutkan berdasarkan skor kecocokan (default).

  Jika parameter `sortOrder` ada tetapi `sortBy` tidak ada, hasilnya diurutkan berdasarkan `score` dalam urutan yang ditentukan oleh `sortOrder`.

- `id`— Urutkan berdasarkan ID, yang berarti URI subjek SPARQL atau simpul Gremlin atau ID tepi.

  Sebagai contoh, dalam kueri SPARQL, Anda dapat menentukan:

  ```
      neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_id' .
  ```

  Dalam kueri Gremlin yang sama, Anda bisa menentukan:

  ```
      .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_id')
  ```

- `label` — Urutkan berdasarkan label.

  Sebagai contoh, dalam kueri SPARQL, Anda dapat menentukan:

  ```
      neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_type' .
  ```

  Dalam kueri Gremlin yang sama, Anda bisa menentukan:

  ```
      .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_type')
  ```

- `doc_type` — Urutkan berdasarkan jenis dokumen (yaitu SPARQL atau Gremlin).

  Sebagai contoh, dalam kueri SPARQL, Anda dapat menentukan:

  ```
      neptune-fts:config neptune-fts:sortBy 'Neptune#fts.document_type' .
  ```

Dalam kueri Gremlin yang sama, Anda bisa menentukan:

```
.withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.document_type')
```

Secara default, OpenSearch hasil tidak diurutkan dan urutannya non-deterministik, artinya kueri yang sama dapat mengembalikan item dalam urutan yang berbeda setiap kali dijalankan. Untuk alasan ini, jika set hasil lebih besar dari `max_result_window`, subset yang sangat berbeda dari hasil total dapat dikembalikan setiap kali kueri dijalankan. Namun, dengan mengurutkan, Anda dapat membuat hasil dari proses yang berbeda lebih dapat dibandingkan secara langsung.

Jika tidak ada parameter `sortOrder` yang menyertai `sortBy`, urutan turun (`DESC`) dari terbesar ke terkecil digunakan.

- **`sortOrder`**— Parameter opsional yang memungkinkan Anda menentukan apakah OpenSearch hasil diurutkan dari terkecil ke terbesar atau dari terbesar ke yang paling kecil (default):

  - `ASC` — Urutan naik, dari terkecil ke terbesar.
  - `DESC` — Urutan turun, dari yang terbesar ke terkecil.

    Ini adalah nilai default, digunakan ketika parameter `sortBy` ada tetapi `sortOrder` tidak ditentukan.

  Jika tidak ada `sortBy` atau tidak `sortOrder` ada, OpenSearch hasil tidak diurutkan secara default.

OpenSearch Pengindeksan non-string di Amazon Neptunus

OpenSearch Pengindeksan non-string di Amazon Neptunus memungkinkan replikasi nilai non-string untuk predikat untuk menggunakan poller aliran. OpenSearch Semua nilai predikat yang dapat dengan aman dikonversi ke OpenSearch pemetaan atau tipe data yang sesuai kemudian direplikasi ke. OpenSearch

Agar pengindeksan non-string diaktifkan pada tumpukan baru, `Enable Non-String Indexing` bendera di CloudFormation template harus disetel ke. `true` Ini adalah pengaturan default. Untuk memperbarui tumpukan yang ada untuk mendukung pengindeksan non-string, lihat Memperbarui tumpukan yang ada di bawah.

> **Note**
>
> - Cara terbaik adalah tidak mengaktifkan pengindeksan non-string pada versi mesin lebih awal dari. **`1.0.4.2`**
> - OpenSearch kueri menggunakan ekspresi reguler untuk nama bidang yang cocok dengan beberapa bidang, beberapa di antaranya berisi nilai string dan yang lainnya berisi nilai non-string, gagal dengan kesalahan. Hal yang sama terjadi jika kueri pencarian teks lengkap di Neptunus adalah jenis itu.
> - Saat mengurutkan berdasarkan bidang non-string, tambahkan ".value" ke nama bidang untuk membedakannya dari bidang string.

Daftar Isi

- Memperbarui tumpukan pencarian teks lengkap Neptunus yang ada untuk mendukung pengindeksan non-string
- Memfilter bidang apa yang diindeks dalam pencarian teks lengkap Neptunus
  - Filter berdasarkan properti atau nama predikat
  - Filter berdasarkan properti atau jenis nilai predikat
- Pemetaan tipe data SPARQL dan Gremlin ke OpenSearch
- Validasi pemetaan data
- Contoh OpenSearch kueri non-string di Neptunus
  - Dapatkan semua simpul dengan usia lebih dari 30 dan nama dimulai dengan "Si"
  - Dapatkan semua node dengan usia antara 10 dan 50 dan nama dengan kecocokan fuzzy dengan "Ronka"
  - Dapatkan semua node dengan stempel waktu yang jatuh dalam 25 hari terakhir
  - Dapatkan semua node dengan stempel waktu yang termasuk dalam tahun dan bulan tertentu

Memperbarui tumpukan pencarian teks lengkap Neptunus yang ada untuk mendukung pengindeksan non-string

Jika Anda sudah menggunakan pencarian teks lengkap Neptunus, berikut adalah langkah-langkah yang perlu Anda ambil untuk mendukung pengindeksan non-string:

1. Hentikan fungsi Lambda poller aliran. Ini memastikan bahwa tidak ada pembaruan baru yang disalin selama ekspor. Lakukan ini dengan menonaktifkan aturan peristiwa cloud yang memanggil fungsi Lambda:

   - Di Konsol Manajemen AWS, navigasikan ke CloudWatch.
   - Pilih Aturan.
   - Pilih aturan dengan nama poller aliran Lambda.
   - Pilih nonaktifkan untuk menonaktifkan aturan sementara.

2. Hapus indeks Neptunus saat ini di. OpenSearch Gunakan `curl` kueri berikut untuk menghapus `amazon_neptune` indeks dari OpenSearch klaster Anda:

   ```
   curl -X DELETE "your OpenSearch endpoint/amazon_neptune"
   ```

3. Mulai ekspor satu kali dari Neptunus ke. OpenSearch Yang terbaik adalah menyiapkan OpenSearch tumpukan baru pada saat ini, sehingga artefak baru diambil untuk poller yang melakukan ekspor.

   Ikuti langkah-langkah yang tercantum di sini GitHub untuk memulai ekspor satu kali data Neptunus Anda. OpenSearch

4. Perbarui artefak Lambda untuk poller aliran yang ada. Setelah ekspor data Neptunus OpenSearch berhasil diselesaikan, lakukan langkah-langkah berikut:

   - Di Konsol Manajemen AWS, navigasikan ke CloudFormation.
   - Pilih CloudFormation tumpukan induk utama.
   - Pilih opsi Perbarui untuk tumpukan itu.
   - Pilih Ganti template saat ini dari opsi.
   - Untuk sumber template, pilih URL Amazon S3.
   - Untuk URL Amazon S3, masukkan:

     ```
     https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
     neptune_to_elastic_search.json
     ```

   - Pilih Berikutnya tanpa mengubah CloudFormation parameter apa pun.
   - Pilih Perbarui tumpukan. CloudFormation akan mengganti artefak kode Lambda untuk poller aliran dengan artefak terbaru.

5. Mulai poller aliran lagi. Lakukan ini dengan mengaktifkan CloudWatch aturan yang sesuai:

- Di Konsol Manajemen AWS, navigasikan ke CloudWatch.
- Pilih Aturan.
- Pilih aturan dengan nama poller aliran Lambda.
- Pilih aktifkan.

Memfilter bidang apa yang diindeks dalam pencarian teks lengkap Neptunus

Ada dua bidang dalam detail CloudFormation template yang memungkinkan Anda menentukan properti atau kunci predikat atau tipe data untuk dikecualikan dari pengindeksan: OpenSearch

Filter berdasarkan properti atau nama predikat

Anda dapat menggunakan parameter CloudFormation template opsional bernama `Properties to exclude from being inserted into Elastic Search Index` untuk menyediakan daftar properti atau kunci predikat yang dibatasi koma untuk dikecualikan dari pengindeksan. OpenSearch

Misalnya, Anda menyetel parameter ini ke`bob`:

```
"Properties to exclude from being inserted into Elastic Search Index" : bob
```

Dalam hal ini, catatan aliran dari kueri pembaruan Gremlin berikut akan dihapus daripada masuk ke indeks:

```
g.V("1").property("bob", "test")
```

Demikian pula, Anda dapat mengatur parameter ke`http://my/example#bob`:

```
"Properties to exclude from being inserted into Elastic Search Index" : http://my/
example#bob
```

Dalam hal ini, catatan aliran kueri pembaruan SPARQL berikut akan dihapus daripada masuk ke indeks:

```
PREFIX ex: <http://my/example#>
INSERT DATA { ex:s1 ex:bob "test"}.
```

Jika Anda tidak memasukkan apa pun dalam parameter CloudFormation template ini, semua kunci properti yang tidak dikecualikan akan diindeks.

Filter berdasarkan properti atau jenis nilai predikat

Anda dapat menggunakan parameter CloudFormation template opsional bernama `Datatypes to exclude from being inserted into Elastic Search Index` untuk menyediakan daftar properti yang dibatasi koma atau tipe data nilai predikat untuk dikecualikan dari pengindeksan. OpenSearch

Untuk SPARQL, Anda tidak perlu mencantumkan URI tipe XSD lengkap, Anda cukup mencantumkan token tipe data. Token tipe data yang valid yang dapat Anda daftarkan adalah:

- `string`
- `boolean`
- `float`
- `double`
- `dateTime`
- `date`
- `time`
- `byte`
- `short`
- `int`
- `long`
- `decimal`
- `integer`
- `nonNegativeInteger`
- `nonPositiveInteger`
- `negativeInteger`
- `unsignedByte`
- `unsignedShort`
- `unsignedInt`
- `unsignedLong`

Untuk Gremlin, tipe data yang valid untuk dicantumkan adalah:

- `string`
- `date`
- `bool`
- `byte`
- `short`
- `int`
- `long`
- `float`
- `double`

Misalnya, Anda menyetel parameter ini ke`string`:

```
"Datatypes to exclude from being inserted into Elastic Search Index" : string
```

Dalam hal ini, catatan aliran dari kueri pembaruan Gremlin berikut akan dihapus daripada masuk ke indeks:

```
g.V("1").property("myStringval", "testvalue")
```

Demikian pula, Anda dapat mengatur parameter ke`int`:

```
"Datatypes to exclude from being inserted into Elastic Search Index" : int
```

Dalam hal ini, catatan aliran kueri pembaruan SPARQL berikut akan dihapus daripada masuk ke indeks:

```
PREFIX ex: <http://my/example#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
INSERT DATA { ex:s1 ex:bob "11"^^xsd:int }.
```

Jika Anda tidak memasukkan apa pun dalam parameter CloudFormation template ini, semua properti yang nilainya dapat dikonversi dengan aman ke OpenSearch ekuivalen akan diindeks. Jenis terdaftar yang tidak didukung oleh bahasa kueri diabaikan.

Pemetaan tipe data SPARQL dan Gremlin ke OpenSearch

Pemetaan tipe data baru dibuat berdasarkan tipe data OpenSearch yang digunakan di properti atau objek. Karena beberapa bidang berisi nilai dari jenis yang berbeda, pemetaan awal mungkin mengecualikan beberapa nilai bidang.

Tipe data Neptunus memetakan ke tipe data sebagai berikut: OpenSearch

Jenis SPARQL	Jenis Gremlin	OpenSearch jenis
`XSD:int`	`byte`	`long`
`XSD:unsignedInt`	`short`	
`XSD:integer`	`int`	
`XSD:byte`	`long`	
`XSD:unsignedByte`		
`XSD:short`		
`XSD:unsignedShort`		
`XSD:long`		
`XSD:unsignedLong`		
`XSD:float`	`float`	`double`
`XSD:double`	`double`	
`XSD:decimal`		
`XSD:boolean`	`bool`	`boolean`
`XSD:datetime`	`date`	`date`
`XSD:date`		
`XSD:string`	`string`	`text`

Jenis SPARQL	Jenis Gremlin	OpenSearch jenis
`XSD:time`		
Jenis data khusus	N/A	`text`
Tipe data lainnya	N/A	`text`

Misalnya, kueri pembaruan Gremlin berikut menyebabkan pemetaan baru untuk “NewField” ditambahkan, yaitu: OpenSearch `{ "type" : "double" }`

```
g.V("1").property("newField" 10.5)
```

Demikian pula, kueri pembaruan SPARQL berikut menyebabkan pemetaan baru untuk “ex:byte” ditambahkan, yaitu: OpenSearch `{ "type" : "long" }`

```
PREFIX ex: <http://my/example#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>

INSERT DATA { ex:test ex:byte "123"^^xsd:byte }.
```

> **Note**
>
> Seperti yang Anda lihat, item yang dipetakan dari Neptunus OpenSearch ke mungkin berakhir dengan tipe data yang berbeda dari yang ada di OpenSearch Neptunus. Namun, ada bidang teks eksplisit di OpenSearch, “datatype”, yang mencatat tipe data yang dimiliki item tersebut di Neptunus.

Validasi pemetaan data

Data direplikasi OpenSearch dari Neptunus menggunakan proses ini:

- Jika pemetaan untuk bidang yang dimaksud sudah ada di OpenSearch:
  - Jika data dapat dikonversi dengan aman ke pemetaan yang ada menggunakan aturan validasi data, maka simpan bidang tersebut. OpenSearch
  - Jika tidak, jatuhkan catatan pembaruan aliran yang sesuai.

- Jika tidak ada pemetaan yang ada untuk bidang yang dimaksud, temukan OpenSearch tipe data yang sesuai dengan tipe data bidang di Neptunus.
  - Jika data bidang dapat dikonversi dengan aman ke OpenSearch tipe data menggunakan aturan validasi data, maka simpan data pemetaan dan bidang baru di. OpenSearch
  - Jika tidak, jatuhkan catatan pembaruan aliran yang sesuai.

Nilai divalidasi terhadap OpenSearch tipe yang setara atau OpenSearch pemetaan yang ada daripada tipe Neptunus. Misalnya, validasi untuk nilai `"123"` dalam `"123"^^xsd:int` dilakukan terhadap `long` tipe daripada `int` tipe.

Meskipun Neptunus mencoba mereplikasi semua data OpenSearch, ada kasus di mana tipe data sama sekali berbeda dari yang ada OpenSearch di Neptunus, dan dalam kasus seperti itu catatan dilewati daripada diindeks. OpenSearch

Misalnya, di Neptunus satu properti dapat memiliki beberapa nilai dari jenis yang berbeda, sedangkan OpenSearch di bidang harus memiliki tipe yang sama di seluruh indeks.

Dengan mengaktifkan log debug, Anda dapat melihat catatan apa yang telah dijatuhkan selama ekspor dari Neptunus ke. OpenSearch Contoh entri log debug adalah:

```
Dropping Record : Data type not a valid Gremlin type
<Record>
```

Tipe data divalidasi sebagai berikut:

- **`text`**— Semua nilai di Neptunus dapat dengan aman dipetakan ke teks. OpenSearch
- **`long`**— Aturan berikut untuk tipe data Neptunus berlaku ketika OpenSearch tipe pemetaan panjang (dalam contoh di bawah ini, diasumsikan memiliki tipe pemetaan): `"testLong"` `long`
  - `boolean`— Tidak valid, tidak dapat dikonversi, dan catatan pembaruan aliran yang sesuai dijatuhkan.

    Contoh Gremlin yang tidak valid adalah:

    ```
    "testLong" : true.
    "testLong" : false.
    ```

    Contoh SPARQL yang tidak valid adalah:

```
":testLong" : "true"^^xsd:boolean
":testLong" : "false"^^xsd:boolean
```

- `datetime`— Tidak valid, tidak dapat dikonversi, dan catatan pembaruan aliran yang sesuai dijatuhkan.

  Contoh Gremlin yang tidak valid adalah:

  ```
  ":testLong" :  datetime('2018-11-04T00:00:00').
  ```

  Contoh SPARQL yang tidak valid adalah:

  ```
  ":testLong" : "2016-01-01"^^xsd:date
  ```

- `float`,`double`, atau `decimal` — Jika nilai di Neptunus adalah bilangan bulat yang dapat muat dalam 64 bit, itu valid dan disimpan OpenSearch dalam panjang, tetapi jika memiliki bagian pecahan, atau adalah a atau, atau lebih besar dari 9.223.372.036.854.775.807 atau lebih kecil dari -9.223.372.372.036,854.775.808, maka tidak valid dan catatan pembaruan aliran yang sesuai dijatuhkan. `NaN` `INF`

  Contoh Gremlin yang valid adalah:

  ```
  "testLong" :  145.0.
  ":testLong" :  123
  ":testLong" :  -9223372036854775807
  ```

  Contoh SPARQL yang valid adalah:

  ```
  ":testLong" : "145.0"^^xsd:float
  ":testLong" :  145.0
  ":testLong" : "145.0"^^xsd:double
  ":testLong" : "145.0"^^xsd:decimal
  ":testLong" : "-92233720368547758O7"
  ```

  Contoh Gremlin yang tidak valid adalah:

  ```
  "testLong" :  123.45
  ":testLong" :  9223372036854775900
  ```

Contoh SPARQL yang tidak valid adalah:

```
":testLong" :  123.45
":testLong" :  9223372036854775900
":testLong" : "123.45"^^xsd:float
":testLong" : "123.45"^^xsd:double
":testLong" : "123.45"^^xsd:decimal
```

- `string`— Jika nilai di Neptunus adalah representasi string dari bilangan bulat yang dapat terkandung dalam integer 64-bit, maka itu valid dan dikonversi ke dalam. `long` OpenSearch Nilai string lainnya tidak valid untuk `long` pemetaan Elasticseearch, dan catatan pembaruan aliran yang sesuai dihapus.

  Contoh Gremlin yang valid adalah:

  ```
  "testLong" :  "123".
  ":testLong" :  "145.0"
  ":testLong" :  "-9223372036854775807"
  ```

  Contoh SPARQL yang valid adalah:

  ```
  ":testLong" : "145.0"^^xsd:string
  ":testLong" : "-9223372036854775807"^^xsd:string
  ```

  Contoh Gremlin yang tidak valid adalah:

  ```
  "testLong" :  "123.45"
  ":testLong" :  "9223372036854775900"
  ":testLong" :  "abc"
  ```

  Contoh SPARQL yang tidak valid adalah:

  ```
  ":testLong" : "123.45"^^xsd:string
  ":testLong" : "abc"
  ":testLong" : "9223372036854775900"^^xsd:string
  ```

- **`double`**— Jika jenis OpenSearch pemetaannya`double`, aturan berikut berlaku (di sini, bidang “testDouble” diasumsikan memiliki `double` pemetaan): OpenSearch

- `boolean`— Tidak valid, tidak dapat dikonversi, dan catatan pembaruan aliran yang sesuai dijatuhkan.

  Contoh Gremlin yang tidak valid adalah:

  ```
    "testDouble" : true.
    "testDouble" : false.
  ```

  Contoh SPARQL yang tidak valid adalah:

  ```
    ":testDouble" : "true"^^xsd:boolean
    ":testDouble" : "false"^^xsd:boolean
  ```

- `datetime`— Tidak valid, tidak dapat dikonversi, dan catatan pembaruan aliran yang sesuai dijatuhkan.

  Contoh Gremlin yang tidak valid adalah:

  ```
    ":testDouble" :  datetime('2018-11-04T00:00:00').
  ```

  Contoh SPARQL yang tidak valid adalah:

  ```
    ":testDouble" : "2016-01-01"^^xsd:date
  ```

- Floating-point `NaN` atau `INF` — Jika nilai di SPARQL adalah floating-point `NaN` atau`INF`, maka itu tidak valid dan catatan pembaruan aliran yang sesuai dijatuhkan.

  Contoh SPARQL yang tidak valid adalah:

  ```
  "  :testDouble" : "NaN"^^xsd:float
    ":testDouble" : "NaN"^^double
    ":testDouble" : "INF"^^double
    ":testDouble" : "-INF"^^double
  ```

- bilangan atau string numerik — Jika nilai di Neptunus adalah bilangan lain atau representasi string numerik dari numnber yang dapat dinyatakan dengan aman sebagai a, maka nilai tersebut valid dan dikonversi menjadi `double` a in. `double` OpenSearch Nilai string lainnya tidak valid untuk OpenSearch `double` pemetaan, dan catatan pembaruan aliran yang sesuai dijatuhkan.

  Contoh Gremlin yang valid adalah:

```
"testDouble" :  123
":testDouble" :  "123"
":testDouble" :  145.67
":testDouble" :  "145.67"
```

Contoh SPARQL yang valid adalah:

```
":testDouble" :  123.45
":testDouble" :  145.0
":testDouble" : "123.45"^^xsd:float
":testDouble" : "123.45"^^xsd:double
":testDouble" : "123.45"^^xsd:decimal
":testDouble" : "123.45"^^xsd:string
```

Contoh Gremlin yang tidak valid adalah:

```
":testDouble" :  "abc"
```

Contoh SPARQL yang tidak valid adalah:

```
":testDouble" : "abc"
```

- **`date`**— Jika jenis OpenSearch pemetaannya`date`, `date` Neptunus `dateTime` dan nilainya valid, seperti halnya nilai string apa pun yang berhasil diuraikan ke format. `dateTime`

  Contoh yang valid di Gremlin atau SPARQL adalah:

```
Date(2016-01-01)
"2016-01-01" "
2003-09-25T10:49:41"
"2003-09-25T10:49"
"2003-09-25T10"
"20030925T104941-0300"
"20030925T104941"
"2003-Sep-25" "
Sep-25-2003"
"2003.Sep.25"
"2003/09/25"
"2003 Sep 25" "
Wed, July 10, '96"
```

```
  "Tuesday, April 12, 1952 AD 3:30:42pm PST"
  "123"
  "-123"
  "0"
  "-0"
  "123.00"
  "-123.00"
```

Contoh yang tidak valid adalah:

```
  123.45
  True
  "abc"
```

Contoh OpenSearch kueri non-string di Neptunus

Neptunus saat ini tidak OpenSearch mendukung kueri jangkauan secara langsung. Namun, Anda dapat mencapai efek yang sama menggunakan sintaks Lucene dan query-type="query_string”, seperti yang dapat Anda lihat dalam contoh kueri berikut.

Dapatkan semua simpul dengan usia lebih dari 30 dan nama dimulai dengan “Si”

Di Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.age.value:>30 && predicates.name.value:Si*');
```

Di SPARQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*age.value:>30 AND
 predicates.\\*foaf\\*name.value:Si*" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
```

```
  }
}
```

Di sini, "*foaf*age digunakan sebagai pengganti URI penuh untuk singkatnya. Ekspresi reguler ini akan mengambil semua bidang memiliki keduanya foaf dan age dalam URI.

Dapatkan semua node dengan usia antara 10 dan 50 dan nama dengan kecocokan fuzzy dengan “Ronka”

Di Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.age.value:[10 TO 50] AND
 predicates.name.value:Ronka~');
```

Di SPARQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*age.value:[10 TO 50] AND
 predicates.\\*foaf\\*name.value:Ronka~" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Dapatkan semua node dengan stempel waktu yang jatuh dalam 25 hari terakhir

Di Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:>now-25d');
```

Di SPARQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\
\*timestamp.value:>now-25d~" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Dapatkan semua node dengan stempel waktu yang termasuk dalam tahun dan bulan tertentu

Di Gremlin, menggunakan ekspresi matematika tanggal dalam sintaks Lucene, untuk Desember 2020:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:>2020-12');
```

Alternatif Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:[2020-12 TO 2021-01]');
```

Full-text-search eksekusi kueri di Amazon Neptunus

Dalam kueri yang menyertakan full-text-search, Neptunus mencoba untuk menempatkan panggilan terlebih dahulu, sebelum bagian lain full-text-search dari kueri. Ini mengurangi jumlah panggilan ke OpenSearch dan dalam banyak kasus secara signifikan meningkatkan kinerja. Namun, ini sama sekali bukan hard-and-fast aturan. Ada situasi, misalnya, di mana `PatternNode` atau `UnionNode` dapat mendahului panggilan pencarian teks lengkap.

Pertimbangkan kueri Gremlin berikut ke basis data yang berisi 100.000 instans `Person`:

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .hasLabel('Person')
 .has('name', 'Neptune#fts marcello~');
```

Jika kueri ini dijalankan dalam urutan di mana langkah-langkah muncul maka 100.000 solusi akan mengalir ke OpenSearch, menyebabkan ratusan OpenSearch panggilan. Faktanya, Neptunus OpenSearch memanggil terlebih dahulu dan kemudian bergabung dengan hasil Neptunus. Dalam kebanyakan kasus, ini jauh lebih cepat daripada mengeksekusi kueri dalam urutan asli.

Anda dapat mencegah pengurutan ulang eksekusi langkah kueri ini menggunakan petunjuk kueri noReordering:

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .withSideEffect('Neptune#noReordering', true)
 .hasLabel('Person')
 .has('name', 'Neptune#fts marcello~');
```

Dalam kasus kedua ini, langkah `.hasLabel` dijalankan terlebih dulu dan langkah `.has('name', 'Neptune#fts marcello~')` setelahnya.

Untuk contoh lain, pertimbangkan kueri SPARQL terhadap jenis data yang sama:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT ?person WHERE {
  ?person rdf:type foaf:Person .
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mike' .
    neptune-fts:config neptune-fts:return ?person .
  }
}
```

Di sini sekali lagi, Neptune mengeksekusi bagian `SERVICE` dari kueri terlebih dulu, dan kemudian menggabungkan hasil dengan data `Person`. Anda dapat menekan perilaku ini menggunakan petunjuk kueri joinOrder:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?person WHERE {
  hint:Query hint:joinOrder "Ordered" .
  ?person rdf:type foaf:Person .
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mike' .
    neptune-fts:config neptune-fts:return ?person .
  }
}
```

Sekali lagi, dalam kueri kedua, bagian-bagian dieksekusi dalam urutan kemunculan mereka dalam kueri.

> **Note**
>
> Menanyakan alias opensearch melalui indeks, alih-alih secara langsung menanyakan indeks opensearch dapat menghasilkan hasil yang salah. Anda harus menanyakan indeks opensearch secara langsung dan bukan alias.

Contoh kueri SPARQL menggunakan pencarian teks lengkap di Neptunus

Berikut ini adalah beberapa contoh kueri SPARQL yang menggunakan pencarian teks lengkap di Amazon Neptune.

Contoh kueri pencocokan SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'match' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'michael' .
    neptune-fts:config neptune-fts:return ?res .
```

```
  }
}
```

Contoh kueri awalan SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'prefix' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mich' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Contoh kueri fuzzy SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'fuzzy' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mikael' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Contoh kueri istilah SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'term' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'Dr. Kunal' .
```

```
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Contoh kueri query_string SPARQL

Kueri ini menentukan beberapa bidang.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ OR rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:field foaf:surname .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Contoh kueri SPARQL simple_query_string

Kueri berikut menentukan bidang menggunakan karakter wildcard ('*').

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'simple_query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

SPARQL urutkan berdasarkan contoh kueri bidang string

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy foaf:name .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

SPARQL urutkan berdasarkan contoh kueri bidang non-string

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name.value .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy dc:date.value .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

SPARQL urutkan berdasarkan contoh kueri ID

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_id' .
    neptune-fts:config neptune-fts:return ?res .
```

```
  }
}
```

SPARQL urutkan berdasarkan contoh kueri label

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_type' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

SPARQL urutkan berdasarkan contoh kueri doc_type

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.document_type' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Contoh menggunakan sintaks Lucene di SPARQL

Sintaks Lucene hanya didukung untuk `query_string` kueri di. OpenSearch

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'predicates.\\foaf\\name.value:micheal AND
 predicates.\\foaf\\surname.value:sh' .
    neptune-fts:config neptune-fts:field ‘’ .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Menggunakan pencarian teks lengkap Neptunus dalam kueri Gremlin

`NeptuneSearchStep` memungkinkan kueri pencarian teks lengkap untuk bagian dari traversal Gremlin yang tidak diubah menjadi langkah-langkah Neptune. Contohnya, pertimbangkan kueri berikut ini.

```
g.withSideEffect("Neptune#fts.endpoint", "your-es-endpoint-URL")
  .V()
      .tail(100)
      .has("name", "Neptune#fts mark*")            <== # Limit the search on name
```

Kueri ini diubah menjadi traversal yang dioptimalkan berikut di Neptune.

```
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=4}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [NeptuneTailGlobalStep(100),
 NeptuneTinkerpopTraverserConverterStep, NeptuneSearchStep {
    JoinGroupNode {
```

```
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=your-OpenSearch-endpoint-URL}
    }
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=your-OpenSearch-endpoint-URL}
    }
}]
```

Contoh berikut adalah dari kueri Gremlin terhadap data rute udara:

Kueri dasar case-insensitive Gremlin **`match`**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'match')
  .V().has("city","Neptune#fts dallas")

==>v[186]
==>v[8]
```

Permintaan Gremlin **`match`**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'match')
  .V().has("city","Neptune#fts southampton")
     .local(values('code','city').fold())
     .limit(5)

==>[SOU, Southampton]
```

Permintaan Gremlin **`fuzzy`**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has("city","Neptune#fts allas~").values('city').limit(5)

==>Dallas
==>Dallas
```

```
==>Walla Walla
==>Velas
==>Altai
```

Permintaan fuzzy Gremlin **query_string**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has("city","Neptune#fts allas~").values('city').limit(5)

==>Dallas
==>Dallas
```

Kueri ekspresi **query_string** reguler Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has("city","Neptune#fts /[dp]allas/").values('city').limit(5)

==>Dallas
==>Dallas
```

Kueri hibrida Gremlin

Kueri ini menggunakan indeks internal Neptunus dan indeks dalam kueri OpenSearch yang sama.

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has("region","GB-ENG")
      .has('city','Neptune#fts L*')
      .values('city')
      .dedup()
      .limit(10)

==>London
==>Leeds
==>Liverpool
==>Land's End
```

Contoh pencarian teks lengkap Gremlin sederhana

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has('desc','Neptune#fts regional municipal')
      .local(values('code','desc').fold())
      .limit(100)

==>[HYA, Barnstable Municipal Boardman Polando Field]
==>[SPS, Sheppard Air Force Base-Wichita Falls Municipal Airport]
==>[ABR, Aberdeen Regional Airport]
==>[SLK, Adirondack Regional Airport]
==>[BFD, Bradford Regional Airport]
==>[EAR, Kearney Regional Airport]
==>[ROT, Rotorua Regional Airport]
==>[YHD, Dryden Regional Airport]
==>[TEX, Telluride Regional Airport]
==>[WOL, Illawarra Regional Airport]
==>[TUP, Tupelo Regional Airport]
==>[COU, Columbia Regional Airport]
==>[MHK, Manhattan Regional Airport]
==>[BJI, Bemidji Regional Airport]
==>[HAS, Hail Regional Airport]
==>[ALO, Waterloo Regional Airport]
==>[SHV, Shreveport Regional Airport]
==>[ABI, Abilene Regional Airport]
==>[GIZ, Jizan Regional Airport]
==>[USA, Concord Regional Airport]
==>[JMS, Jamestown Regional Airport]
==>[COS, City of Colorado Springs Municipal Airport]
==>[PKB, Mid Ohio Valley Regional Airport]
```

Kueri Gremlin menggunakan **`query_string`** Dengan Operator '+' dan '-'

Meskipun jenis kueri `query_string` jauh lebih pemaaf dari jenis `simple_query_string` default, kueri tersebut memungkinkan kueri yang lebih tepat. Kueri pertama di bawah ini menggunakan `query_string`, sedangkan yang kedua menggunakan `simple_query_string` default:

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
 . V().has('desc','Neptune#fts +London -(Stansted|Gatwick)')
```

```
      .local(values('code','desc').fold())
      .limit(10)

==>[LHR, London Heathrow]
==>[YXU, London Airport]
==>[LTN, London Luton Airport]
==>[SEN, London Southend Airport]
==>[LCY, London City Airport]
```

Perhatikan bagaimana `simple_query_string` dalam contoh di bawah diam-diam mengabaikan operator '+' dan '-':

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has('desc','Neptune#fts +London -(Stansted|Gatwick)')
      .local(values('code','desc').fold())
      .limit(10)

==>[LHR, London Heathrow]
==>[YXU, London Airport]
==>[LGW, London Gatwick]
==>[STN, London Stansted Airport]
==>[LTN, London Luton Airport]
==>[SEN, London Southend Airport]
==>[LCY, London City Airport]
==>[SKG, Thessaloniki Macedonia International Airport]
==>[ADB, Adnan Menderes International Airport]
==>[BTV, Burlington International Airport]
```

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has('desc','Neptune#fts +(regional|municipal) -(international|bradford)')
      .local(values('code','desc').fold())
      .limit(10)

==>[CZH, Corozal Municipal Airport]
==>[MMU, Morristown Municipal Airport]
==>[YBR, Brandon Municipal Airport]
==>[RDD, Redding Municipal Airport]
==>[VIS, Visalia Municipal Airport]
==>[AIA, Alliance Municipal Airport]
==>[CDR, Chadron Municipal Airport]
```

```
==>[CVN, Clovis Municipal Airport]
==>[SDY, Sidney Richland Municipal Airport]
==>[SGU, St George Municipal Airport]
```

query_string Permintaan Gremlin dengan **AND** dan operator **OR**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has('desc','Neptune#fts (St AND George) OR (St AND Augustin)')
      .local(values('code','desc').fold())
      .limit(10)

==>[YIF, St Augustin Airport]
==>[STG, St George Airport]
==>[SGO, St George Airport]
==>[SGU, St George Municipal Airport]
```

Permintaan Gremlin **term**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'term')
  .V().has("SKU","Neptune#fts ABC123DEF9")
      .local(values('code','city').fold())
      .limit(5)

==>[AUS, Austin]
```

Permintaan Gremlin **prefix**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'prefix')
  .V().has("icao","Neptune#fts ka")
      .local(values('code','icao','city').fold())
      .limit(5)

==>[AZO, KAZO, Kalamazoo]
==>[APN, KAPN, Alpena]
```

```
==>[ACK, KACK, Nantucket]
==>[ALO, KALO, Waterloo]
==>[ABI, KABI, Abilene]
```

Menggunakan sintaks Lucene di Neptunus Gremlin

Di Neptune Gremlin, Anda juga dapat menulis kueri yang sangat kuat menggunakan sintaks kueri Lucene. Perhatikan bahwa sintaks Lucene hanya didukung untuk `query_string` kueri di. OpenSearch

Asumsikan data berikut:

```
g.addV("person")
        .property(T.id, "p1")
        .property("name", "simone")
        .property("surname", "rondelli")

g.addV("person")
        .property(T.id, "p2")
        .property("name", "simone")
        .property("surname", "sengupta")

g.addV("developer")
        .property(T.id, "p3")
        .property("name", "simone")
        .property("surname", "rondelli")
```

Menggunakan sintaks Lucene, yang dipanggil ketika `queryType` adalah `query_string`, Anda dapat mencari data ini menurut nama dan nama keluarga sebagai berikut:

```
g.withSideEffect("Neptune#fts.endpoint", "es_endpoint")
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts predicates.name.value:simone AND
 predicates.surname.value:rondelli")

==> v[p1], v[p3]
```

Perhatikan bahwa dalam langkah `has()` di atas, bidangnya digantikan oleh `"*"`). Sebenarnya, setiap nilai yang ditempatkan di bidang tersebut diganti oleh bidang yang Anda akses dalam kueri.

Anda mengakses bidang nama menggunakan `predicates.name.value,` karena itulah cara model data terstruktur.

Anda dapat mencari berdasarkan nama, nama keluarga, dan label, sebagai berikut:

```
g.withSideEffect("Neptune#fts.endpoint", getEsEndpoint())
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts predicates.name.value:simone AND
 predicates.surname.value:rondelli AND entity_type:person")

==> v[p1]
```

Label diakses menggunakan `entity_type`, sekali lagi karena itulah cara model data terstruktur.

Anda juga bisa memasukkan syarat nesting:

```
g.withSideEffect("Neptune#fts.endpoint", getEsEndpoint())
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts (predicates.name.value:simone AND
 predicates.surname.value:rondelli AND entity_type:person) OR
 predicates.surname.value:sengupta")

==> v[p1], v[p2]
```

Memasukkan grafik modern TinkerPop

```
g.addV('person').property(T.id, '1').property('name', 'marko').property('age', 29)
 .addV('personr').property(T.id, '2').property('name', 'vadas').property('age', 27)
 .addV('software').property(T.id, '3').property('name', 'lop').property('lang', 'java')
 .addV('person').property(T.id, '4').property('name', 'josh').property('age', 32)
 .addV('software').property(T.id, '5').property('name', 'ripple').property('lang',
 'java')
 .addV('person').property(T.id, '6').property('name', 'peter').property('age', 35)

g.V('1').as('a').V('2').as('b').addE('knows').from('a').to('b').property('weight',
 0.5f).property(T.id, '7')
 .V('1').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.4f).property(T.id, '9')
 .V('4').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.4f).property(T.id, '11')
```

```
 .V('4').as('a').V('5').as('b').addE('created').from('a').to('b').property('weight',
 1.0f).property(T.id, '10')
 .V('6').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.2f).property(T.id, '12')
 .V('1').as('a').V('4').as('b').addE('knows').from('a').to('b').property('weight',
 1.0f).property(T.id, '8')
```

Urutkan berdasarkan contoh nilai bidang string

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'name')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Urutkan berdasarkan contoh nilai bidang non-string

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'age.value')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Urutkan berdasarkan contoh nilai bidang ID

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
.withSideEffect('Neptune#fts.queryType', 'query_string')
.withSideEffect('Neptune#fts.sortOrder', 'asc')
.withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_id')
.V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Urutkan berdasarkan contoh nilai bidang label

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_type')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Urutkan berdasarkan contoh nilai **`document_type`** bidang

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.document_type')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Memecahkan masalah pencarian teks lengkap Neptunus

> **Note**
>
> Jika Anda telah mengaktifkan kontrol akses berbutir halus di OpenSearch cluster Anda, Anda perlu mengaktifkan otentikasi IAM di database Neptunus Anda juga.

Untuk mendiagnosis masalah dengan replikasi dari Neptunus OpenSearch ke, lihat Log untuk fungsi Lambda CloudWatch poller Anda. Log ini memberikan rincian tentang jumlah catatan yang dibaca dari aliran dan jumlah catatan yang berhasil direplikasi. OpenSearch

Anda juga dapat mengubah tingkat LOGGING untuk fungsi Lambda Anda dengan mengubah variabel lingkungan `LoggingLevel`.

> **Note**
>
> Dengan `LoggingLevel` set to`DEBUG`, Anda dapat melihat detail tambahan, seperti catatan aliran yang dijatuhkan dan alasan mengapa masing-masing dijatuhkan, sambil mereplikasi data StreamPoller dari Neptunus ke. OpenSearch Ini dapat berguna jika Anda menemukan Anda kehilangan catatan.

Aplikasi konsumen streaming Neptunus menerbitkan dua metrik yang juga dapat membantu Anda mendiagnosis CloudWatch masalah:

- `StreamRecordsProcessed` — Jumlah catatan yang diproses oleh aplikasi per satuan waktu. Bermanfaat dalam melacak tingkat run aplikasi.

- `StreamLagTime` — Perbedaan waktu dalam milidetik antara waktu saat ini dan waktu komit catatan aliran sedang diproses. Metrik ini menunjukkan berapa banyak aplikasi konsumen yang tertinggal.

Selain itu, semua metrik yang terkait dengan proses replikasi diekspos di dasbor di CloudWatch bawah nama yang sama dengan yang `ApplicationName` disediakan saat Anda membuat instance aplikasi menggunakan templat. CloudWatch

Anda juga dapat memilih untuk membuat CloudWatch alarm yang dipicu setiap kali polling gagal lebih dari dua kali berturut-turut. Lakukan ini dengan menetapkan bidang `CreateCloudWatchAlarm` ke `true` ketika Anda menginisiasi aplikasi. Setelah itu, tentukan alamat email yang ingin Anda beri tahu saat alarm dipicu.

Pemecahan masalah proses yang gagal saat membaca catatan dari aliran

Jika proses gagal saat membaca catatan dari aliran, pastikan bahwa Anda memiliki hal berikut:

- Aliran diaktifkan pada klaster Anda.
- Titik akhir aliran Neptune menggunakan format yang benar:
  - Untuk Gremlin atau OpenCypher: atau aliasnya, `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/propertygraph/stream` `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/pg/stream`
  - Untuk SPARQL: `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/sparql/stream`
- Titik akhir DynamoDB dikonfigurasi untuk VPC Anda.
- Titik akhir pemantauan dikonfigurasi untuk subnet VPC Anda.

Memecahkan masalah proses yang gagal saat menulis data OpenSearch

Jika proses gagal saat menulis catatan OpenSearch, pastikan Anda memiliki yang berikut ini:

- Versi Elasticsearch Anda adalah 7.1 atau lebih tinggi, atau Opensearch 2.3 dan di atasnya.
- OpenSearch dapat diakses dari fungsi poller Lambda di VPC Anda.
- Kebijakan keamanan yang dilampirkan untuk OpenSearch memungkinkan HTTP/HTTPS permintaan masuk.

Memperbaiki out-of-sync masalah antara Neptunus OpenSearch dan pengaturan replikasi yang ada

Anda dapat menggunakan langkah-langkah di bawah ini untuk mendapatkan database OpenSearch dan domain Neptunus kembali sinkron dengan data terbaru jika terjadi out-of-sync masalah di antara mereka yang disebabkan oleh korupsi `ExpiredStreamException` atau data.

Perhatikan bahwa pendekatan ini menghapus semua data dalam OpenSearch domain dan menyinkronkannya kembali dari status database Neptunus saat ini, jadi tidak ada data yang perlu dimuat ulang dalam database Neptunus.

1. Nonaktifkan proses replikasi seperti yang dijelaskan dalam Menonaktifkan (menjeda) proses poller aliran.
2. Hapus indeks Neptunus pada domain menggunakan OpenSearch perintah berikut:

   ```
   curl -X DELETE "(your OpenSearch endpoint)/amazon_neptune"
   ```

3. Buat klon database (lihat Kloning Database di Neptunus).
4. Dapatkan yang terbaru `eventID` untuk aliran pada database kloning dengan menjalankan perintah semacam ini terhadap titik akhir Streams API (lihat Memanggil REST API Neptunus Streams untuk informasi selengkapnya):

   ```
   curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
   iteratorType=LATEST"
   ```

   Catat nilai-nilai di `commitNum` dan `opNum` bidang di `lastEventId` objek dalam respons.
5. Gunakan export-neptune-to-elasticsearchalat di github untuk melakukan sinkronisasi satu kali dari database kloning ke domain. OpenSearch
6. Pergi ke tabel DynamoDB untuk tumpukan replikasi. Nama tabel akan menjadi Nama Aplikasi yang Anda tentukan dalam CloudFormation template (defaultnya adalah`NeptuneStream`) dengan `-LeaseTable` akhiran. Dengan kata lain, nama tabel default adalah`NeptuneStream-LeaseTable`.

   Anda dapat menjelajahi baris tabel dengan memindai karena seharusnya hanya ada satu baris dalam tabel. Buat perubahan berikut menggunakan `commitNum` dan `opNum` nilai yang Anda rekam di atas:

   - Ubah nilai untuk `checkpoint` bidang dalam tabel ke nilai yang Anda catat`commitNum`.

- Ubah nilai untuk `checkpointSubSequenceNumber` bidang dalam tabel ke nilai yang Anda catatopNum.

7. Aktifkan kembali proses replikasi seperti yang dijelaskan dalam Mengaktifkan kembali proses poller aliran.
8. Hapus database kloning dan CloudFormation tumpukan yang dibuat untuk `export-neptune-to-elasticsearch` alat ini.

Menggunakan AWS Lambda fungsi di Amazon Neptunus

AWS Lambda fungsi memiliki banyak kegunaan dalam aplikasi Amazon Neptunus. Di sini kami memberikan panduan umum untuk menggunakan fungsi Lambda dengan salah satu driver Gremlin populer dan varian bahasa, dan contoh spesifik fungsi Lambda yang ditulis dalam Java,, dan Python. JavaScript

> **Note**
>
> Cara terbaik untuk menggunakan fungsi Lambda dengan Neptune telah diubah dengan rilis mesin baru-baru ini. Neptune yang digunakan untuk meninggalkan koneksi idle terbuka lama setelah konteks eksekusi Lambda telah didaur ulang, berpotensi menyebabkan kebocoran sumber daya pada server. Untuk mengurangi ini, kami merekomendasikan membuka dan menutup koneksi dengan setiap invokasi Lambda. Dimulai dengan mesin versi 1.0.3.0, namun, batas waktu koneksi siaga telah dikurangi sehingga koneksi tidak lagi bocor setelah konteks eksekusi Lambda tidak aktif telah didaur ulang, sehingga sekarang kami sarankan untuk menggunakan koneksi tunggal selama durasi konteks eksekusi. Ini harus mencakup beberapa penanganan kesalahan dan kode back-off-and-retry boilerplate untuk menangani koneksi yang ditutup secara tidak terduga.

Mengelola WebSocket koneksi Gremlin dalam fungsi AWS Lambda

Jika Anda menggunakan varian bahasa Gremlin untuk menanyakan Neptunus, driver terhubung ke database menggunakan koneksi. WebSocket WebSockets dirancang untuk mendukung skenario koneksi client-server yang berumur panjang. AWS Lambda, di sisi lain, dirancang untuk mendukung eksekusi yang relatif berumur pendek dan tanpa kewarganegaraan. Ketidakcocokan dalam filosofi desain dapat menyebabkan beberapa masalah tak terduga saat menggunakan Lambda untuk kueri Neptune.

AWS Lambda Fungsi berjalan dalam konteks eksekusi yang mengisolasi fungsi dari fungsi lain. Konteks eksekusi dibuat pertama kalinya saat fungsi dipanggil dan dapat digunakan kembali untuk invokasi berikutnya dari fungsi yang sama.

Namun, salah satu konteks eksekusi tidak pernah digunakan untuk menangani beberapa invokasi yang bersamaan dari fungsi. Jika fungsi Anda dipanggil secara bersamaan oleh beberapa klien,

Lambda memutar sebuah konteks eksekusi tambahan untuk setiap instans dari fungsi. Semua konteks eksekusi baru ini pada gilirannya dapat digunakan kembali untuk invokasi berikutnya dari fungsi.

Pada titik tertentu, Lambda mendaur ulang konteks eksekusi, terutama jika mereka tidak aktif selama beberapa waktu. AWS Lambda mengekspos siklus hidup konteks eksekusi, termasuk, `Invoke` dan `Shutdown` fase`Init`, melalui ekstensi Lambda. Menggunakan ekstensi ini, Anda dapat menulis kode yang membersihkan sumber daya eksternal seperti koneksi basis data ketika konteks eksekusi di daur ulang.

Praktik terbaik yang umum adalah membuka koneksi basis data di luar fungsi handler Lambda sehingga dapat digunakan kembali dengan setiap panggilan handler. Jika koneksi basis data turun di beberapa titik, Anda dapat menyambung kembali dari dalam handler. Namun, ada bahaya kebocoran koneksi dengan pendekatan ini. Jika koneksi idle tetap terbuka lama setelah konteks eksekusi hancur, skenario invokasi Lambda berterusan atau berselang secara bertahap dapat membocorkan koneksi dan menghabiskan sumber daya basis data.

Batas koneksi Neptune dan timeout koneksi telah berubah dengan rilis mesin yang lebih baru. Sebelumnya, setiap instance mendukung hingga 60.000 WebSocket koneksi. Sekarang, jumlah maksimum WebSocket koneksi bersamaan per instance Neptunus bervariasi dengan jenis instance.

Juga, dimulai dengan rilis mesin 1.0.3.0, Neptune mengurangi batas waktu idle untuk koneksi dari satu jam ke sekitar 20 menit. Jika klien tidak menutup koneksi, koneksi ditutup secara otomatis setelah batas waktu idle 20 hingga 25 menit. AWS Lambda tidak mendokumentasikan masa pakai konteks eksekusi, tetapi eksperimen menunjukkan bahwa batas waktu koneksi Neptunus baru sejalan dengan batas waktu konteks eksekusi Lambda yang tidak aktif. Pada saat konteks eksekusi tidak aktif didaur ulang, ada kemungkinan koneksinya telah ditutup oleh Neptune, atau akan ditutup segera setelah itu.

Rekomendasi untuk digunakan AWS Lambda dengan Amazon Neptunus Gremlin

Kami sekarang merekomendasikan menggunakan koneksi tunggal dan sumber grafik traversal untuk seumur hidup dari konteks eksekusi Lambda, bukan satu untuk setiap invokasi fungsi (setiap fungsi invkasi hanya menangani hanya satu permintaan klien). Karena permintaan klien yang bersamaan ditangani oleh instans fungsi yang berbeda yang berjalan di dalam konteks eksekusi terpisah, maka tidak perlu mempertahankan kolam koneksi untuk menangani permintaan yang bersamaan dalam

instans fungsi. Jika driver Gremlin yang Anda gunakan memiliki kolam koneksi, konfigurasikan driver tersebut untuk menggunakan hanya satu koneksi.

Untuk menangani kegagalan koneksi, gunakan logika coba lagi di sekitar masing-masing kueri. Meskipun tujuannya adalah untuk mempertahankan koneksi tunggal untuk seumur hidup konteks eksekusi, peristiwa jaringan tak terduga dapat menyebabkan koneksi tersebut dihentikan tiba-tiba. Kegagalan koneksi seperti itu terwujud sebagai kesalahan yang berbeda tergantung pada driver yang Anda gunakan. Anda harus melakukan kode fungsi Lambda Anda untuk menangani masalah koneksi ini dan mencoba koneksi ulang jika diperlukan.

Beberapa driver Gremlin secara otomatis menangani koneksi ulang. Driver Java, misalnya, secara otomatis mencoba untuk membangun kembali konektivitas ke Neptune atas nama kode klien Anda. Dengan driver ini, kode fungsi Anda hanya perlu mundur dan coba lagi kueri. Driver JavaScript dan Python, sebaliknya, tidak menerapkan logika koneksi ulang otomatis apa pun, jadi dengan driver ini kode fungsi Anda harus mencoba menyambung kembali setelah mundur, dan hanya mencoba lagi kueri setelah koneksi dibuat kembali.

Contoh kode di sini termasuk logika rekoneksi daripada menganggap bahwa klien mengurus itu.

Rekomendasi untuk menggunakan permintaan tulis Gremlin di Lambda

Jika fungsi Lambda Anda memodifikasi data grafik, pertimbangkan untuk mengadopsi back-off-and-retry strategi untuk menangani pengecualian berikut:

- **`ConcurrentModificationException`** – Semantik transaksi Neptune berarti bahwa permintaan tulis terkadang gagal dengan `ConcurrentModificationException`. Dalam situasi ini, cobalah mekanisme coba lagi back-off-based eksponensial.
- **`ReadOnlyViolationException`** – Karena topologi klaster dapat berubah setiap saat sebagai akibat dari peristiwa yang direncanakan atau tidak direncanakan, menulis tanggung jawab dapat bermigrasi dari satu instans ke lainnya dalam klaster. Jika kode fungsi Anda mencoba untuk mengirim permintaan tulis ke instans yang bukan lagi instans utama (penulis), permintaan gagal dengan `ReadOnlyViolationException`. Ketika ini terjadi, tutup koneksi yang ada, sambung kembali ke titik akhir klaster, dan kemudian coba lagi permintaannya.

Selain itu, jika Anda menggunakan back-off-and-retry strategi untuk menangani masalah permintaan tulis, pertimbangkan untuk menerapkan kueri idempoten untuk membuat dan memperbarui permintaan (misalnya, menggunakan fold () .coalesce () .unfold ().

Rekomendasi untuk menggunakan permintaan tulis Gremlin di Lambda

Jika Anda memiliki satu replika baca atau lebih klaster Anda, itu ide yang baik untuk menyeimbangkan permintaan baca di seluruh replika ini. Salah satu pilihan adalah dengan menggunakan reader endpoint. Reader Endpoint menyeimbangkan koneksi di seluruh replika bahkan jika topologi klaster berubah ketika Anda menambahkan atau menghapus replika, atau mempromosikan replika untuk menjadi instans primer baru.

Namun, menggunakan Reader Endpoint dapat mengakibatkan penggunaan sumber daya klaster yang tidak merata dalam beberapa keadaan. Reader Endpoint bekerja dengan mengubah host yang ditunjuk entri DNS secara berkala. Jika klien membuka banyak koneksi sebelum perubahan entri DNS, semua permintaan koneksi dikirim ke instans Neptune tunggal. Hal ini dapat menjadi kasus dengan skenario Lambda throughput tinggi di mana sejumlah besar permintaan bersamaan untuk fungsi Lambda Anda menyebabkan beberapa konteks eksekusi akan dibuat, masing-masing dengan koneksinya sendiri. Jika koneksi tersebut semuanya dibuat hampir bersamaan, koneksi cenderung ke semua titik pada replika yang sama dalam klaster, dan untuk tetap menunjuk ke replika itu sampai konteks eksekusi didaur ulang.

Salah satu cara Anda dapat mendistribusikan permintaan di seluruh instans adalah untuk mengkonfigurasi fungsi Lambda Anda agar terhubung ke titik akhir instans, dipilih secara acak dari daftar titik akhir instans replika, bukan Reader Endpoint. Kelemahan dari pendekatan ini adalah bahwa pendekatan ini memerlukan kode Lambda untuk menangani perubahan dalam topologi klaster dengan memantau klaster dan memperbarui daftar titik akhir setiap kali keanggotaan klaster berubah.

Jika Anda menulis fungsi Lambda Java yang perlu menyeimbangkan permintaan baca di seluruh instans di klaster Anda, Anda dapat menggunakan Klien Gremlin untuk Amazon Neptune, klien Gremlin Java yang mengetahui topologi klaster Anda dan yang dengan adil mendistribusikan koneksi dan permintaan di seluruh satu set instans di klaster Neptune. Posting blog ini termasuk contoh fungsi Lambda Java yang menggunakan klien Gremlin untuk Amazon Neptune.

Faktor-faktor yang dapat memperlambat awal dingin dari fungsi Lambda Gremlin Neptune

Pertama kali AWS Lambda fungsi dipanggil disebut sebagai awal yang dingin. Ada beberapa faktor yang dapat meningkatkan latensi awal dingin:

- Pastikan untuk menetapkan memori yang cukup untuk fungsi Lambda Anda. — Kompilasi selama start dingin dapat secara signifikan lebih lambat untuk fungsi Lambda daripada yang

akan diaktifkan EC2 karena AWS Lambda mengalokasikan siklus CPU secara linier sebanding dengan memori yang Anda tetapkan ke fungsi tersebut. Dengan memori 1.769 MB, sebuah fungsi menerima setara dengan satu vCPU penuh (satu VCPU-detik kredit per detik). Dampak tidak menetapkan cukup memori untuk menerima siklus CPU yang memadai yakni terutama ditentukan untuk fungsi Lambda besar yang tertulis di Java.

- Sadarilah bahwa mengaktifkan autentikasi basis data IAM dapat memperlambat awal dingin – AWS Identity and Access Management (IAM) autentikasi database juga dapat memperlambat awal dingin, terutama jika fungsi Lambda untuk menghasilkan kunci penandatanganan baru. Latency ini hanya mempengaruhi awal dingin dan bukan permintaan berikutnya, karena setelah IAM DB auth telah menetapkan kredensial koneksi, Neptune hanya secara berkala memvalidasi bahwa mereka masih berlaku.

Menggunakan CloudFormation untuk Membuat Fungsi Lambda untuk Digunakan di Neptunus

Anda dapat menggunakan CloudFormation template untuk membuat AWS Lambda fungsi yang dapat mengakses Neptunus.

1. Untuk meluncurkan tumpukan fungsi Lambda di CloudFormation konsol, pilih salah satu tombol Launch Stack di tabel berikut.

Region	Lihat	Lihat di Designer	Luncurkan
US East (N. Virginia)	Lihat	Lihat di Desainer	
AS Timur (Ohio)	Lihat	Lihat di Desainer	
US West (N. California)	Lihat	Lihat di Desainer	
US West (Oregon)	Lihat	Lihat di Desainer	
Kanada (Pusat)	Lihat	Lihat di Desainer	

Region	Lihat	Lihat di Designer	Luncurkan
Amerika Selatan (Sao Paulo)	Lihat	Lihat di Desainer	
Europe (Stockholm)	Lihat	Lihat di Desainer	
Eropa (Irlandia)	Lihat	Lihat di Desainer	
Eropa (London)	Lihat	Lihat di Desainer	
Europe (Paris)	Lihat	Lihat di Desainer	
Eropa (Frankfurt)	Lihat	Lihat di Desainer	
Timur Tengah (Bahrain)	Lihat	Lihat di Desainer	
Timur Tengah (UAE)	Lihat	Lihat di Desainer	
Israel (Tel Aviv)	Lihat	Lihat di Desainer	
Africa (Cape Town)	Lihat	Lihat di Desainer	
Asia Pasifik (Hong Kong)	Lihat	Lihat di Desainer	
Asia Pacific (Tokyo)	Lihat	Lihat di Desainer	
Asia Pasifik (Seoul)	Lihat	Lihat di Desainer	
Asia Pacific (Singapore)	Lihat	Lihat di Desainer	

Region	Lihat	Lihat di Designer	Luncurkan
Asia Pacific (Sydney)	Lihat	Lihat di Desainer	
Asia Pasifik (Mumbai)	Lihat	Lihat di Desainer	
Tiongkok (Beijing)	Lihat	Lihat di Desainer	
Tiongkok (Ningxia)	Lihat	Lihat di Desainer	
AWS GovCloud (AS-Barat)	Lihat	Lihat di Desainer	
AWS GovCloud (AS-Timur)	Lihat	Lihat di Desainer	

2. Pada halaman Pilih Templat, pilih Selanjutnya.
3. Di halaman Tentukan Detail, lihat opsi berikut:

   a. Pilih waktu aktif Lambda, tergantung pada bahasa apa yang ingin Anda gunakan dalam fungsi Lambda Anda. CloudFormation Template ini saat ini mendukung bahasa-bahasa berikut:

      - Python 3.9 (peta ke URL Amazon `python39` S3)
      - NodeJS 18 (peta `nodejs18x` ke URL Amazon S3)
      - Ruby 2.5 (peta untuk `ruby25` dalam URL Amazon S3)

   b. Menyediakan titik akhir klaster Neptune dan nomor port yang sesuai.

   c. Menyediakan grup keamanan Neptune yang sesuai.

   d. Menyediakan parameter subnet Neptune yang sesuai.
4. Pilih Berikutnya.
5. Pada halaman Opsi, pilih Selanjutnya.
6. Pada halaman Review, pilih kotak centang pertama untuk mengetahui bahwa CloudFormation akan membuat sumber daya IAM.

   Lalu pilih Buat.

Jika Anda perlu membuat perubahan sendiri pada waktu aktif Lambda, Anda dapat men-download generik dari lokasi Amazon S3 di Wilayah Anda:

```
https://s3.Amazon region.amazonaws.com/aws-neptune-customer-samples-Amazon region/
lambda/runtime-language/lambda_function.zip.
```

Contoh:

```
https://s3.us-west-2.amazonaws.com/aws-neptune-customer-samples-us-west-2/lambda/
python36/lambda_function.zip
```

AWS Lambda contoh fungsi untuk Amazon Neptunus

Contoh AWS Lambda fungsi berikut, ditulis dalam Java, JavaScript dan Python, menggambarkan upserting satu simpul dengan ID yang dihasilkan secara acak menggunakan idiom. `fold().coalesce().unfold()`

Banyak kode di setiap fungsi adalah kode boilerplate, bertanggung jawab untuk mengelola koneksi dan mencoba kembali koneksi juga query jika terjadi kesalahan. Logika aplikasi nyata dan query Gremlin diimplementasikan dalam metode `doQuery()` dan`query()` berturut-turut. Jika Anda menggunakan contoh-contoh ini sebagai dasar untuk fungsi Lambda Anda sendiri, Anda dapat berkonsentrasi pada modifikasi `doQuery()` dan `query()`.

Fungsi dikonfigurasi untuk mencoba lagi kueri yang gagal 5 kali, menunggu 1 detik antara pengulangan itu.

Fungsi memerlukan nilai untuk hadir dalam variabel lingkungan Lambda berikut:

- **`NEPTUNE_ENDPOINT`** – Titik akhir klaster DB Neptune Anda. Untuk Python, ini seharusnya. `neptuneEndpoint`
- **`NEPTUNE_PORT`** – Port Neptune. Untuk Python, ini seharusnya. `neptunePort`
- **`USE_IAM`** — (`true`ataufalse) Jika database Anda memiliki otentikasi database AWS Identity and Access Management (IAM) diaktifkan, atur variabel `USE_IAM` lingkungan ke. `true` Hal ini menyebabkan fungsi Lambda untuk permintaan koneksi SIGV4-sign ke Neptune. Untuk permintaan auth DB IAM seperti itu, pastikan bahwa peran eksekusi fungsi Lambda memiliki kebijakan IAM tepat yang terpasang, yang memungkinkan fungsi untuk terhubung ke klaster Neptune DB Anda (lihat Jenis kebijakan IAM).

Contoh Fungsi Lambda Java untuk Amazon Neptune

Berikut adalah beberapa hal yang perlu diingat tentang AWS Lambda fungsi Java:

- Driver Java mempertahankan kolam koneksinya sendiri, yang tidak Anda butuhkan, jadi konfigurasikan objek `Cluster` dengan `minConnectionPoolSize(1)` dan `maxConnectionPoolSize(1)`.
- Objek `Cluster` bisa saja lambat untuk dibangun karena objek tersebut membuat satu atau lebih serializers (Gyro secara default, ditambah lainnya jika Anda telah mengkonfigurasinya untuk format output tambahan seperti `binary`). Ini perlu beberapa waktu untuk instantiate.
- Kolam koneksi diinisialisasi dengan permintaan pertama. Pada titik ini, driver mengatur tumpukan `Netty`, mengalokasikan byte buffer, dan membuat kunci penandatanganan jika Anda menggunakan IAM DB auth. Semua yang dapat menambah latensi cold-start.
- Kolam koneksi Driver Java memonitor ketersediaan host server dan secara otomatis mencoba menyambung kembali jika koneksi gagal. Itu dimulai tugas latar belakang untuk mencoba membangun kembali koneksi. Gunakan `reconnectInterval( )` untuk mengkonfigurasi interval antara upaya rekoneksi. Sementara driver sedang mencoba untuk menyambung kembali, fungsi Lambda Anda hanya dapat mencoba lagi query.

  Jika interval antara mencoba kembali lebih kecil daripada interval antara upaya menyambung kembali, mencoba kembali pada koneksi yang akan gagal lagi karena host dianggap tidak tersedia. Ini tidak berlaku untuk mencoba lagi `ConcurrentModificationException`.
- Gunakan Java 8 bukan Java 11. Optimasi `Netty` tidak diaktifkan secara default di Java 11.
- Contoh ini menggunakan Retry4j untuk mencoba ulang.
- Untuk menggunakan penandatanganan driver `Sigv4` dalam fungsi Lambda Java Anda, lihat persyaratan dependensi di Menghubungkan ke database Amazon Neptunus menggunakan IAM dengan Gremlin Java.

> **Warning**
>
> `CallExecutor`Dari Retry4j mungkin tidak aman untuk utas. Pertimbangkan agar setiap utas menggunakan `CallExecutor` instance-nya sendiri.

> **Note**
>
> Contoh berikut telah diperbarui untuk menyertakan penggunaan requestInterceptor (). Ini ditambahkan di TinkerPop 3.6.6. Sebelum TinkerPop versi 3.6.6, contoh kode menggunakan HandshakeInterceptor (), yang tidak digunakan lagi dengan rilis itu.

```
package com.amazonaws.examples.social;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestStreamHandler;
import com.evanlennick.retry4j.CallExecutor;
import com.evanlennick.retry4j.CallExecutorBuilder;
import com.evanlennick.retry4j.Status;
import com.evanlennick.retry4j.config.RetryConfig;
import com.evanlennick.retry4j.config.RetryConfigBuilder;
import org.apache.tinkerpop.gremlin.driver.Cluster;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer;
import org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteConnection;
import org.apache.tinkerpop.gremlin.driver.ser.Serializers;
import org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource;
import org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversalSource;
import org.apache.tinkerpop.gremlin.structure.T;

import java.io.*;
import java.time.temporal.ChronoUnit;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;
import java.util.concurrent.Callable;
import java.util.function.Function;

import static java.nio.charset.StandardCharsets.UTF_8;
import static org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.__.addV;
import static org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.__.unfold;

public class MyHandler implements RequestStreamHandler {

  private final GraphTraversalSource g;
  private final CallExecutor<Object> executor;
  private final Random idGenerator = new Random();
```

```
public MyHandler() {

  this.g = AnonymousTraversalSource
          .traversal()
          .withRemote(DriverRemoteConnection.using(createCluster()));


  this.executor = new CallExecutorBuilder<Object>()
          .config(createRetryConfig())
          .build();

}

@Override
public void handleRequest(InputStream input,
                          OutputStream output,
                          Context context) throws IOException {

  doQuery(input, output);
}

private void doQuery(InputStream input, OutputStream output) throws IOException {
  try {

    Map<String, Object> args = new HashMap<>();
    args.put("id", idGenerator.nextInt());

    String result = query(args);

    try (Writer writer = new BufferedWriter(new OutputStreamWriter(output, UTF_8))) {
        writer.write(result);
    }

  } finally {
      input.close();
      output.close();
  }
}

private String query(Map<String, Object> args) {
  int id = (int) args.get("id");

  @SuppressWarnings("unchecked")
```

```
    Callable<Object> query = () -> g.V(id)
      .fold()
      .coalesce(
        unfold(),
        addV("Person").property(T.id, id))
      .id().next();

    Status<Object> status = executor.execute(query);

    return status.getResult().toString();
  }

  private Cluster createCluster() {
    Cluster.Builder builder = Cluster.build()

.addContactPoint(System.getenv("NEPTUNE_ENDPOINT"))

.port(Integer.parseInt(System.getenv("NEPTUNE_PORT")))
                                    .enableSsl(true)
                                    .minConnectionPoolSize(1)
                                    .maxConnectionPoolSize(1)
                                    .serializer(Serializers.GRAPHBINARY_V1D0)
                                    .reconnectInterval(2000);

  if (Boolean.parseBoolean(getOptionalEnv("USE_IAM", "true"))) {
    // The following example uses requestInterceptor(), which was introduced
    // in TinkerPop 3.6.6. If you are using a TinkerPop version earlier than
    // 3.6.6 (but 3.5.5 or higher), use handshakeInterceptor() instead.
    builder.requestInterceptor( r ->
      {
        NeptuneNettyHttpSigV4Signer sigV4Signer = new
NeptuneNettyHttpSigV4Signer(region, DefaultCredentialsProvider.create());
        sigV4Signer.signRequest(r);
        return r;
      }
    )

    return builder.create();
  }

  private RetryConfig createRetryConfig() {
    return new RetryConfigBuilder().retryOnCustomExceptionLogic(retryLogic())
                                   .withDelayBetweenTries(1000, ChronoUnit.MILLIS)
                                   .withMaxNumberOfTries(5)
```

```
                                    .withFixedBackoff()
                                    .build();
  }

  private Function<Exception, Boolean> retryLogic() {
    return e -> {
      StringWriter stringWriter = new StringWriter();
      e.printStackTrace(new PrintWriter(stringWriter));
      String message = stringWriter.toString();

      // Check for connection issues
      if ( message.contains("Timed out while waiting for an available host") ||
           message.contains("Timed-out waiting for connection on Host") ||
           message.contains("Connection to server is no longer active") ||
           message.contains("Connection reset by peer") ||
           message.contains("SSLEngine closed already") ||
           message.contains("Pool is shutdown") ||
           message.contains("ExtendedClosedChannelException") ||
           message.contains("Broken pipe")) {
        return true;
      }

      // Concurrent writes can sometimes trigger a ConcurrentModificationException.
      // In these circumstances you may want to backoff and retry.
      if (message.contains("ConcurrentModificationException")) {
          return true;
      }

      // If the primary fails over to a new instance, existing connections to the old
primary will
      // throw a ReadOnlyViolationException. You may want to back and retry.
      if (message.contains("ReadOnlyViolationException")) {
          return true;
      }

      return false;
    };
  }

  private String getOptionalEnv(String name, String defaultValue) {
    String value = System.getenv(name);
    if (value != null && value.length() > 0) {
      return value;
    } else {
```

```
      return defaultValue;
    }
  }
}
```

Jika Anda ingin menyertakan logika sambung kembali dalam fungsi Anda, lihat Sampel sambungan ulang Java.

JavaScript Contoh fungsi Lambda untuk Amazon Neptunus

Catatan tentang contoh ini

- JavaScript Pengemudi tidak memelihara kolam koneksi. Selalu membuka koneksi tunggal.
- Fungsi contoh menggunakan utilitas penandatanganan Sigv4 dari gremlin-aws-sigv4 untuk menandatangani permintaan ke database berkemampuan otentikasi IAM.
- Ini menggunakan fungsi retry () dari modul utilitas async open-source untuk menangani upaya. backoff-and-retry
- Langkah terminal Gremlin mengembalikan a JavaScript `promise` (lihat TinkerPop dokumentasi). Untuk `next()`, ini adalah tupel `{value, done}`.
- Kesalahan koneksi muncul di dalam handler, dan ditangani dengan menggunakan beberapa backoff-and-retry logika sesuai dengan rekomendasi yang diuraikan di sini, dengan satu pengecualian. Ada satu jenis masalah koneksi yang pengemudi tidak perlakukan sebagai pengecualian, dan karena itu tidak dapat diakomodasi oleh backoff-and-retry logika ini.

  Masalahnya adalah jika koneksi ditutup setelah driver mengirimkan permintaan tapi sebelum driver menerima respon, kueri terlihat selesai namun kembali ke nilai null. Sejauh fungsi lambda klien diperhatikan, fungsi terlihat berhasil, tetapi dengan respon kosong.

  Dampak dari masalah ini tergantung pada cara aplikasi Anda memperlakukan respon kosong. Beberapa aplikasi mungkin memperlakukan respon kosong dari permintaan baca sebagai kesalahan, tetapi yang lain mungkin keliru memperlakukannya sebagai hasil kosong.

  Menulis permintaan yang mengalami masalah koneksi ini juga akan mengembalikan respon kosong. Apakah invokasi yang berhasil dengan respon kosong menandakan keberhasilan atau kegagalan? Jika klien yang mengaktifkan fungsi tulis hanya memperlakukan invokasi yang berhasil dari fungsi agar berarti menulis ke database telah dilakukan, daripada memeriksa tubuh respons, sistem mungkin tampak kehilangan data.

Masalah ini keluar dari cara driver memperlakukan peristiwa yang dipancarkan oleh socket yang mendasari. Ketika soket jaringan yang mendasarinya ditutup dengan `ECONNRESET` kesalahan, yang WebSocket digunakan oleh pengemudi ditutup dan memancarkan `'ws close'` peristiwa. Tidak ada apapun dalam driver, namun, untuk menangani peristiwa itu dengan cara yang dapat digunakan untuk meningkatkan pengecualian. Akibatnya, kueri itu hilang begitu saja.

Untuk mengatasi masalah ini, fungsi lambda contoh di sini menambahkan handler peristiwa `'ws close'` yang melempar pengecualian untuk driver saat membuat sambungan jarak jauh. Namun, pengecualian ini tidak muncul di sepanjang jalur permintaan-respons kueri Gremlin, dan karena itu tidak dapat digunakan untuk memicu backoff-and-retry logika apa pun dalam fungsi lambda itu sendiri. Sebaliknya, pengecualian yang dilemparkan oleh handler peristiwa `'ws close'` membawa hasil dalam pengecualian tidak tertangani yang menyebabkan invokasi lambda gagal. Hal ini memungkinkan klien yang meng-invoke fungsi untuk menangani kesalahan dan mencoba lagi invokasi lambda jika sesuai.

Kami menyarankan Anda menerapkan backoff-and-retry logika dalam fungsi lambda itu sendiri untuk melindungi klien Anda dari masalah koneksi intermiten. Namun, pemecahan masalah di atas memerlukan klien agar menerapkan logika coba lagi juga, untuk menangani kegagalan yang dihasilkan dari masalah sambungan tertentu ini.

Kode JavaScript

```
const gremlin = require('gremlin');
const async = require('async');
const {getUrlAndHeaders} = require('gremlin-aws-sigv4/lib/utils');

const traversal = gremlin.process.AnonymousTraversalSource.traversal;
const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;
const t = gremlin.process.t;
const __ = gremlin.process.statics;

let conn = null;
let g = null;

async function query(context) {

  const id = context.id;

  return g.V(id)
```

```
    .fold()
    .coalesce(
      __.unfold(),
      __.addV('User').property(t.id, id)
    )
    .id().next();
}

async function doQuery() {
  const id = Math.floor(Math.random() * 10000).toString();

  let result = await query({id: id});
  return result['value'];
}

exports.handler = async (event, context) => {

  const getConnectionDetails = () => {
    if (process.env['USE_IAM'] == 'true'){
       return getUrlAndHeaders(
         process.env['NEPTUNE_ENDPOINT'],
         process.env['NEPTUNE_PORT'],
         {},
         '/gremlin',
         'wss');
    } else {
      const database_url = 'wss://' + process.env['NEPTUNE_ENDPOINT'] + ':' +
 process.env['NEPTUNE_PORT'] + '/gremlin';
      return { url: database_url, headers: {}};
    }
  };


  const createRemoteConnection = () => {
    const { url, headers } = getConnectionDetails();

    const c = new DriverRemoteConnection(
      url,
      {
        headers: headers
      });

     c._client._connection.on('close', (code, message) => {
         console.info(`close - ${code} ${message}`);
```

```
        if (code == 1006){
          console.error('Connection closed prematurely');
          throw new Error('Connection closed prematurely');
        }
      });

   return c;
};

const createGraphTraversalSource = (conn) => {
  return traversal().withRemote(conn);
};

if (conn == null){
  console.info("Initializing connection")
  conn = createRemoteConnection();
  g = createGraphTraversalSource(conn);
}

return async.retry(
  {
    times: 5,
    interval: 1000,
    errorFilter: function (err) {

      // Add filters here to determine whether error can be retried
      console.warn('Determining whether retriable error: ' + err.message);

      // Check for connection issues
      if (err.message.startsWith('WebSocket is not open')){
        console.warn('Reopening connection');
        conn.close();
        conn = createRemoteConnection();
        g = createGraphTraversalSource(conn);
        return true;
      }

      // Check for ConcurrentModificationException
      if (err.message.includes('ConcurrentModificationException')){
        console.warn('Retrying query because of ConcurrentModificationException');
        return true;
      }

      // Check for ReadOnlyViolationException
```

```
        if (err.message.includes('ReadOnlyViolationException')){
          console.warn('Retrying query because of ReadOnlyViolationException');
          return true;
        }

        return false;
      }

    },
    doQuery);
};
```

Contoh Fungsi Lambda Python untuk Amazon Neptune

Berikut adalah beberapa hal yang perlu diperhatikan tentang fungsi contoh AWS Lambda Python berikut:

- Ini menggunakan modul backoff.
- Ini diatur `pool_size=1` untuk menjaga dari pembuatan kolam koneksi yang tidak perlu.

```
import os, sys, backoff, math
from random import randint
from gremlin_python import statics
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection
from gremlin_python.driver.protocol import GremlinServerError
from gremlin_python.driver import serializer
from gremlin_python.process.anonymous_traversal import traversal
from gremlin_python.process.graph_traversal import __
from gremlin_python.process.strategies import *
from gremlin_python.process.traversal import T
from aiohttp.client_exceptions import ClientConnectorError
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import ReadOnlyCredentials
from types import SimpleNamespace

import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
```

```
reconnectable_err_msgs = [
    'ReadOnlyViolationException',
    'Server disconnected',
    'Connection refused',
    'Connection was already closed',
    'Connection was closed by server',
    'Failed to connect to server: HTTP Error code 403 - Forbidden'
]

retriable_err_msgs = ['ConcurrentModificationException'] + reconnectable_err_msgs

network_errors = [OSError, ClientConnectorError]

retriable_errors = [GremlinServerError, RuntimeError, Exception] + network_errors

def prepare_iamdb_request(database_url):

    service = 'neptune-db'
    method = 'GET'

    access_key = os.environ['AWS_ACCESS_KEY_ID']
    secret_key = os.environ['AWS_SECRET_ACCESS_KEY']
    region = os.environ['AWS_REGION']
    session_token = os.environ['AWS_SESSION_TOKEN']


    creds = SimpleNamespace(
        access_key=access_key, secret_key=secret_key, token=session_token,
 region=region,
    )

    request = AWSRequest(method=method, url=database_url, data=None)
    SigV4Auth(creds, service, region).add_auth(request)

    return database_url, request.headers.items()

def is_retriable_error(e):

    is_retriable = False
    err_msg = str(e)

    if isinstance(e, tuple(network_errors)):
        is_retriable = True
    else:
```

```
        is_retriable = any(retriable_err_msg in err_msg for retriable_err_msg in
 retriable_err_msgs)

    logger.error('error: [{}] {}'.format(type(e), err_msg))
    logger.info('is_retriable: {}'.format(is_retriable))

    return is_retriable

def is_non_retriable_error(e):
    return not is_retriable_error(e)

def reset_connection_if_connection_issue(params):

    is_reconnectable = False

    e = sys.exc_info()[1]
    err_msg = str(e)

    if isinstance(e, tuple(network_errors)):
        is_reconnectable = True
    else:
        is_reconnectable = any(reconnectable_err_msg in err_msg for
 reconnectable_err_msg in reconnectable_err_msgs)

    logger.info('is_reconnectable: {}'.format(is_reconnectable))

    if is_reconnectable:
        global conn
        global g
        conn.close()
        conn = create_remote_connection()
        g = create_graph_traversal_source(conn)

@backoff.on_exception(backoff.constant,
                      tuple(retriable_errors),
                      max_tries=5,
                      jitter=None,
                      giveup=is_non_retriable_error,
                      on_backoff=reset_connection_if_connection_issue,
                      interval=1)
def query(**kwargs):

    id = kwargs['id']
```

```
    return (g.V().hasLabel('column').has('column_name',
 'amhstr_ag_type').in_('hascolumn').dedup().valueMap().limit(10).toList())

def doQuery(event):
    return query(id=str(randint(0, 10000)))

def lambda_handler(event, context):
    result = doQuery(event)
    logger.info('result - {}'.format(result))
    return result

def create_graph_traversal_source(conn):
    return traversal().withRemote(conn)

def create_remote_connection():
    logger.info('Creating remote connection')

    (database_url, headers) = connection_info()

    # Convert headers to a dictionary if it's not already
    headers_dict = dict(headers) if isinstance(headers, list) else headers

    print(headers)
    return DriverRemoteConnection(
        database_url,
        'g',
        pool_size=1,
        headers=headers_dict)

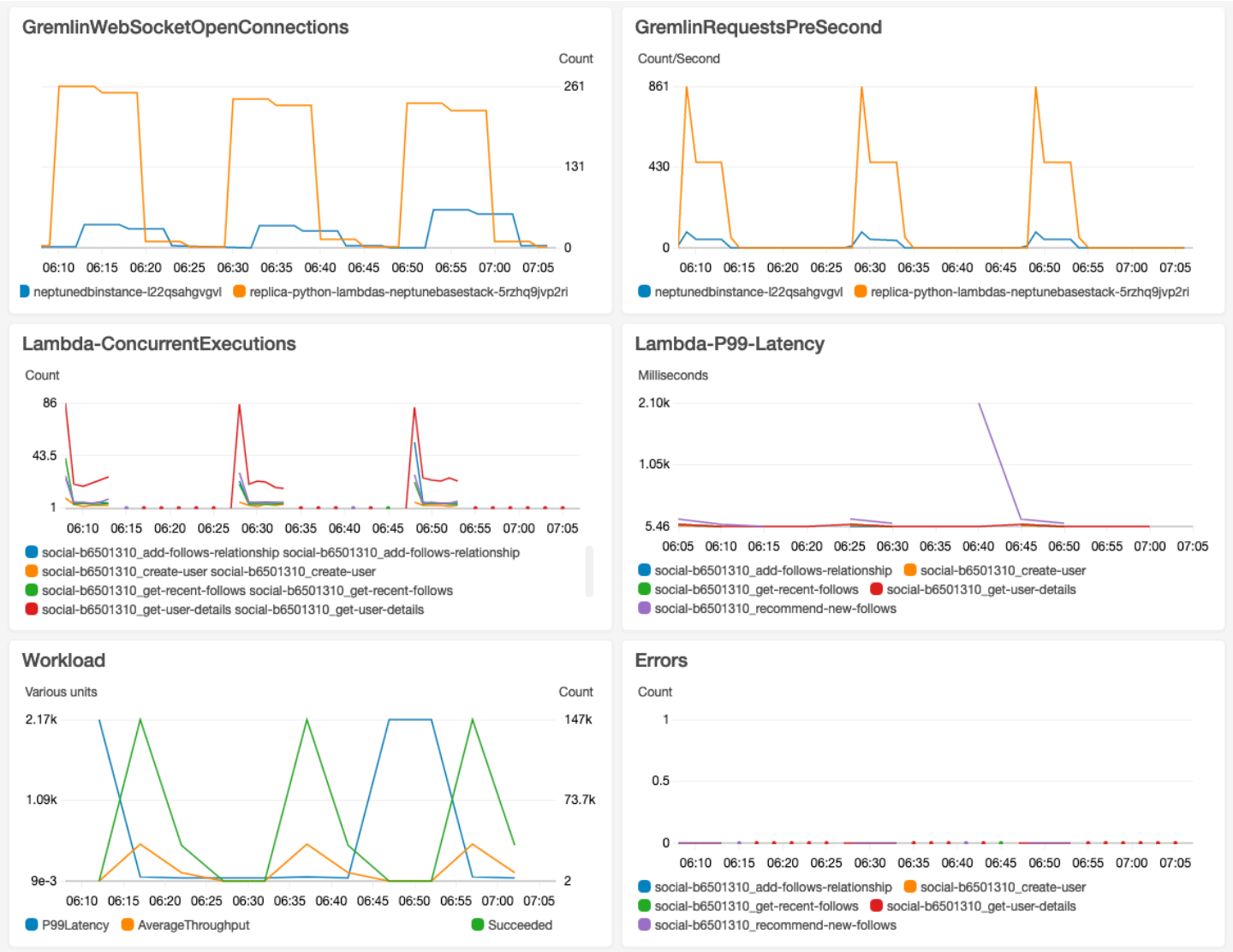

def connection_info():

    database_url = 'wss://{}:{}/gremlin'.format(os.environ['neptuneEndpoint'],
 os.environ['neptunePort'])

    if 'USE_IAM' in os.environ and os.environ['USE_IAM'] == 'true':
        return prepare_iamdb_request(database_url)
    else:
        return (database_url, {})

conn = create_remote_connection()
g = create_graph_traversal_source(conn)
```

Berikut adalah hasil sampel, menunjukkan periode bolak-balik beban berat dan ringan:



Amazon Neptunus ML untuk pembelajaran mesin pada grafik

Sering ada informasi berharga dalam set data terhubung yang besar yang sulit untuk diekstrak menggunakan kueri berdasarkan intuisi manusia saja. Teknik machine learning (ML) dapat membantu menemukan korelasi tersembunyi dalam grafik dengan miliaran hubungan. Korelasi ini sangat membantu untuk merekomendasikan produk, memprediksi kelayakan kredit, mengidentifikasi penipuan, dan banyak hal lainnya.

Fitur Neptune ML memungkinkan untuk membangun dan melatih model machine learning yang berguna pada grafik besar dalam hitungan jam, bukan minggu. Untuk mencapai hal ini, Neptunus ML menggunakan teknologi jaringan saraf grafik (GNN) yang didukung oleh SageMaker Amazon AI dan Deep Graph Library (DGL) (yang merupakan sumber terbuka). Jaringan saraf grafik adalah bidang yang muncul dalam kecerdasan buatan (lihat, misalnya, Survei Komprehensif tentang Jaringan Saraf Grafik). Untuk tutorial langsung tentang penggunaan GNNs dengan DGL, lihat Mempelajari jaringan saraf grafik dengan Deep Graph Library.

> **Note**
>
> Grafik vertex diidentifikasi dalam model Neptune ML sebagai “simpul”. Sebagai contoh, klasifikasi vertex menggunakan model machine learning klasifikasi simpul, dan regresi vertex menggunakan model regresi simpul.

Apa yang bisa dilakukan Neptune ML

Neptunus mendukung inferensi transduktif, yang mengembalikan prediksi yang telah dihitung sebelumnya pada saat pelatihan, berdasarkan data grafik Anda pada waktu itu, dan inferensi induktif, yang mengembalikan menerapkan pemrosesan data dan evaluasi model secara real time, berdasarkan data saat ini. Lihat Perbedaan antara inferensi induktif dan transduktif.

Neptunus ML dapat melatih model pembelajaran mesin untuk mendukung lima kategori inferensi yang berbeda:

Jenis tugas inferensi saat ini didukung oleh Neptune ML

- Klasifikasi simpul — memprediksi fitur kategoris dari properti simpul.

Misalnya, film yang diberikan The Shawshank Redemption, Neptune ML dapat memprediksi properti `genre`-nya sebagai `story` dari satu set kandidat `[story, crime, action, fantasy, drama, family, ...]`.

Ada dua jenis tugas klasifikasi simpul:

- Klasifikasi kelas tunggal: Dalam tugas semacam ini, setiap simpul memiliki hanya satu fitur target. Sebagai contoh, properti, `Place_of_birth` dari `Alan Turing` memiliki nilai UK.
- Klasifikasi kelas ganda: Dalam tugas semacam ini, setiap simpul memiliki lebih dari satu fitur target. Sebagai contoh, properti `genre` dari film The Godfather memiliki nilai `crime`dan `story`.

- Regresi simpul — memprediksi properti numerik dari sebuah simpul.

  Misalnya, film yang diberikan Avengers: Endgame, Neptune ML dapat memprediksi bahwa propertinya `popularity` memiliki nilai `5.0`.

- Klasifikasi tepi — memprediksi fitur kategoris properti tepi.

  Ada dua jenis tugas klasifikasi tepi:

  - Klasifikasi kelas tunggal: Dalam tugas semacam ini, setiap tepi hanya memiliki satu fitur target. Misalnya, tepi peringkat antara pengguna dan film mungkin memiliki properti,`liked`, dengan nilai “Ya” atau “Tidak”.
  - Klasifikasi multi-kelas: Dalam tugas semacam ini, setiap tepi dapat memiliki lebih dari satu fitur target. Misalnya, peringkat antara pengguna dan film mungkin memiliki beberapa nilai untuk tag properti seperti “Lucu”, “Heartwarming”, “Chilling”, dan sebagainya.

- Regresi tepi — memprediksi properti numerik tepi.

  Misalnya, tepi peringkat antara pengguna dan film mungkin memiliki properti numerik`score`, di mana Neptunus ML dapat memprediksi nilai yang diberikan pengguna dan film.

- Prediksi tautan — memprediksi node tujuan yang paling mungkin untuk node sumber tertentu dan tepi keluar, atau node sumber yang paling mungkin untuk node tujuan tertentu dan tepi masuk.

  Misalnya, dengan grafik pengetahuan penyakit obat, diberikan `Aspirin` sebagai simpul sumber, dan `treats` sebagai tepi keluar, Neptunus ML dapat memprediksi node tujuan yang paling relevan sebagai,, dan seterusnya. `heart disease` `fever`

  Atau, dengan grafik pengetahuan Wikimedia, diberikan `President-of` sebagai edge atau relasi dan `United-States` sebagai simpul tujuan, Neptune ML dapat memprediksi kepala yang paling

relevan sebagai `George Washington`, `Abraham Lincoln`, `Franklin D. Roosevelt`, dan sebagainya.

> **Note**
>
> Klasifikasi node dan klasifikasi Edge hanya mendukung nilai string. Itu berarti bahwa nilai properti numerik seperti `0` atau tidak `1` didukung, meskipun string setara `"0"` dan adalah. `"1"` Demikian pula, nilai properti Boolean `true` dan `false` tidak berfungsi, tetapi `"true"` dan `"false"` lakukan.

Dengan Neptune ML, Anda dapat menggunakan model machine learning yang termasuk dalam dua kategori umum:

Jenis model machine learning saat ini didukung oleh Neptune ML

- Model Graph Neural Network (GNN) — Ini termasuk Relational Graph Convolutional Networks (R-). GCNs Model GNN berfungsi untuk ketiga-tiga jenis tugas di atas.
- Model Knowledge-Graph Embedding (KGE) – Ini termasuk model `TransE`, `DistMult`, dan `RotatE`. Mereka hanya sesuai untuk prediksi link.

Model yang ditentukan pengguna - Neptunus ML juga memungkinkan Anda menyediakan implementasi model kustom Anda sendiri untuk semua jenis tugas yang tercantum di atas. Anda dapat menggunakan toolkit Neptunus ML untuk mengembangkan dan menguji implementasi model kustom berbasis python Anda sebelum menggunakan API pelatihan Neptunus Neptunus dengan model Anda. Lihat Model khusus di Neptunus ML detail tentang cara menyusun dan mengatur implementasi Anda sehingga kompatibel dengan infrastruktur pelatihan Neptunus ML.

Menyiapkan Neptunus ML

Cara termudah untuk memulai dengan Neptunus ML adalah dengan menggunakan CloudFormation template quick-start. Template ini menginstal semua komponen yang diperlukan, termasuk cluster DB Neptunus baru, semua peran IAM yang diperlukan, dan notebook grafik Neptunus baru untuk mempermudah bekerja dengan Neptunus ML.

Anda juga dapat menginstal Neptunus ML secara manual, seperti yang dijelaskan di. Menyiapkan Neptunus ML tanpa menggunakan template quick-start CloudFormation

Menggunakan template Neptunus AWS CloudFormation ML untuk memulai dengan cepat di cluster DB baru

Cara termudah untuk memulai dengan Neptunus ML adalah dengan menggunakan CloudFormation template quick-start. Template ini menginstal semua komponen yang diperlukan, termasuk cluster DB Neptunus baru, semua peran IAM yang diperlukan, dan notebook grafik Neptunus baru untuk mempermudah bekerja dengan Neptunus ML.

Untuk membuat tumpukan mulai cepat Neptune ML

1. Untuk meluncurkan CloudFormation tumpukan di CloudFormation konsol, pilih salah satu tombol Launch Stack di tabel berikut:

Wilayah	Lihat	Lihat di Designer	Luncurkan
US East (N. Virginia)	Lihat	Lihat di Desainer	
AS Timur (Ohio)	Lihat	Lihat di Desainer	
US West (N. California)	Lihat	Lihat di Desainer	
US West (Oregon)	Lihat	Lihat di Desainer	
Kanada (Pusat)	Lihat	Lihat di Desainer	
Amerika Selatan (Sao Paulo)	Lihat	Lihat di Desainer	
Europe (Stockholm)	Lihat	Lihat di Desainer	
Eropa (Irlandia)	Lihat	Lihat di Desainer	
Eropa (London)	Lihat	Lihat di Desainer	

Wilayah	Lihat	Lihat di Designer	Luncurkan
Europe (Paris)	Lihat	Lihat di Desainer	
Eropa (Frankfurt)	Lihat	Lihat di Desainer	
Timur Tengah (Bahrain)	Lihat	Lihat di Desainer	
Timur Tengah (UEA)	Lihat	Lihat di Desainer	
Israel (Tel Aviv)	Lihat	Lihat di Desainer	
Afrika (Cape Town)	Lihat	Lihat di Desainer	
Asia Pasifik (Hong Kong)	Lihat	Lihat di Desainer	
Asia Pacific (Tokyo)	Lihat	Lihat di Desainer	
Asia Pasifik (Seoul)	Lihat	Lihat di Desainer	
Asia Pacific (Singapore)	Lihat	Lihat di Desainer	
Asia Pacific (Sydney)	Lihat	Lihat di Desainer	
Asia Pasifik (Mumbai)	Lihat	Lihat di Desainer	
Tiongkok (Beijing)	Lihat	Lihat di Desainer	
Tiongkok (Ningxia)	Lihat	Lihat di Desainer	

Wilayah	Lihat	Lihat di Designer	Luncurkan
AWS GovCloud (AS-Barat)	Lihat	Lihat di Desainer	

2. Pada halaman Pilih Templat, pilih Selanjutnya.
3. Pada halaman Tentukan Detail, pilih Selanjutnya.
4. Pada halaman Opsi, pilih Selanjutnya.
5. Pada halaman Ulasan, ada dua kotak centang yang perlu Anda centang:

   - Yang pertama mengakui bahwa AWS CloudFormation mungkin membuat sumber daya IAM dengan nama khusus.
   - Yang kedua mengakui bahwa AWS CloudFormation mungkin memerlukan `CAPABILITY_AUTO_EXPAND` kemampuan untuk tumpukan baru. `CAPABILITY_AUTO_EXPAND`secara eksplisit memungkinkan CloudFormation untuk memperluas makro secara otomatis saat membuat tumpukan, tanpa tinjauan sebelumnya.

     Pelanggan sering membuat set perubahan dari template yang diproses sehingga perubahan yang dibuat oleh makro dapat ditinjau sebelum benar-benar membuat tumpukan. Untuk informasi selengkapnya, lihat CloudFormation CreateStackAPI.

   Lalu pilih Buat.

Template mulai cepat membuat dan mengatur berikut ini:

- Klaster DB Neptune.
- IAM role yang diperlukan (dan melampirkannya).
- Grup EC2 keamanan Amazon yang diperlukan.
- Titik akhir SageMaker AI VPC yang diperlukan.
- Sebuah grup parameter klaster DB untuk Neptune ML.
- Parameter yang diperlukan dalam grup parameter itu.
- Notebook SageMaker AI dengan sampel notebook yang sudah terisi sebelumnya untuk Neptunus ML. Perhatikan bahwa tidak semua ukuran instans tersedia di setiap wilayah, jadi Anda harus memastikan bahwa ukuran instans notebook yang dipilih adalah salah satu yang didukung wilayah Anda.

- Layanan Ekspor Neptune.

Saat tumpukan mulai cepat siap, buka notebook SageMaker AI yang dibuat template dan lihat contoh yang telah diisi sebelumnya. Mereka akan membantu Anda men-download set data sampel untuk digunakan bereksperimen dengan kemampuan Neptune ML.

Mereka juga dapat menghemat banyak waktu saat Anda menggunakan Neptune ML. Misalnya, lihat sihir %neptune_ml garis, dan sihir %%neptune_ml sel yang didukung notebook ini.

Anda juga dapat menggunakan AWS CLI perintah berikut untuk menjalankan template quick-start CloudFormation :

```
aws cloudformation create-stack \
  --stack-name neptune-ml-fullstack-$(date '+%Y-%m-%d-%H-%M') \
  --template-url https://aws-neptune-customer-samples.s3.amazonaws.com/v2/
cloudformation-templates/neptune-ml-nested-stack.json \
  --parameters ParameterKey=EnableIAMAuthOnExportAPI,ParameterValue=(true if you have
 IAM auth enabled, or false otherwise) \
               ParameterKey=Env,ParameterValue=test$(date '+%H%M')\
  --capabilities CAPABILITY_IAM \
  --region (the AWS region, like us-east-1) \
  --disable-rollback \
  --profile (optionally, a named CLI profile of yours)
```

Menyiapkan Neptunus ML tanpa menggunakan template quick-start CloudFormation

Panduan ini memberikan step-by-step petunjuk untuk menyiapkan Amazon Neptunus ML tanpa menggunakan AWS CloudFormation templat mulai cepat. Ini mengasumsikan Anda sudah memiliki cluster DB Neptunus yang berfungsi dan mencakup pengaturan yang diperlukan, termasuk menginstal layanan Neptunus-Ekspor, membuat peran IAM khusus, dan mengonfigurasi cluster DB Anda untuk mengaktifkan Neptunus ML. Panduan ini juga menjelaskan cara membuat dua titik akhir SageMaker AI di VPC Neptunus Anda untuk memberi mesin Neptunus akses ke manajemen AI yang diperlukan. SageMaker APIs Dengan mengikuti petunjuk ini, Anda dapat mengatur Neptunus ML pada infrastruktur Neptunus yang ada tanpa bergantung pada template. CloudFormation

Mulailah dengan cluster DB Neptunus yang berfungsi

Jika Anda tidak menggunakan template CloudFormation mulai cepat untuk menyiapkan Neptunus ML, Anda memerlukan cluster DB Neptunus yang ada untuk bekerja dengannya. Jika mau, Anda dapat menggunakan yang sudah Anda miliki, atau mengkloning yang sudah Anda gunakan, atau Anda dapat membuat yang baru (lihatBuat cluster Neptunus).

Menginstal layanan Neptune-Ekspor

Jika Anda belum melakukannya, instal layanan Neptune-Export, seperti yang dijelaskan di. Menggunakan layanan Neptunus-Ekspor untuk mengekspor data Neptunus

Tambahkan aturan masuk ke grup `NeptuneExportSecurityGroup` keamanan yang dibuat penginstalan, dengan pengaturan berikut:

- Jenis: `Custom TCP`
- Protokol: `TCP`
- Rentang port: `80 - 443`
- Sumber: *`(Neptune DB cluster security group ID)`*

Buat peran IAM NeptuneLoadFrom S3 kustom

Jika Anda belum melakukannya, buat peran `NeptuneLoadFromS3` IAM khusus, seperti yang dijelaskan diMembuat peran IAM untuk mengakses Amazon S3.

Buat NeptuneSageMaker IAMRole peran khusus

Gunakan konsol IAM untuk membuat kustom`NeptuneSageMakerIAMRole`, menggunakan kebijakan berikut:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:CreateNetworkInterfacePermission",
        "ec2:CreateVpcEndpoint",
        "ec2:DeleteNetworkInterface",
        "ec2:DeleteNetworkInterfacePermission",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
```

```
        "arn:aws:iam::*:role/*"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "sagemaker.amazonaws.com"
          ]
        }
      },
      "Effect": "Allow"
    },
    {
      "Action": [
        "kms:CreateGrant",
        "kms:Decrypt",
        "kms:GenerateDataKey*"
      ],
      "Resource": "arn:aws:kms:*:*:key/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/sagemaker/*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:AddTags",
        "sagemaker:CreateEndpoint",
        "sagemaker:CreateEndpointConfig",
        "sagemaker:CreateHyperParameterTuningJob",
        "sagemaker:CreateModel",
        "sagemaker:CreateProcessingJob",
        "sagemaker:CreateTrainingJob",
        "sagemaker:CreateTransformJob",
```

```
        "sagemaker:DeleteEndpoint",
        "sagemaker:DeleteEndpointConfig",
        "sagemaker:DeleteModel",
        "sagemaker:DescribeEndpoint",
        "sagemaker:DescribeEndpointConfig",
        "sagemaker:DescribeHyperParameterTuningJob",
        "sagemaker:DescribeModel",
        "sagemaker:DescribeProcessingJob",
        "sagemaker:DescribeTrainingJob",
        "sagemaker:DescribeTransformJob",
        "sagemaker:InvokeEndpoint",
        "sagemaker:ListTags",
        "sagemaker:ListTrainingJobsForHyperParameterTuningJob",
        "sagemaker:StopHyperParameterTuningJob",
        "sagemaker:StopProcessingJob",
        "sagemaker:StopTrainingJob",
        "sagemaker:StopTransformJob",
        "sagemaker:UpdateEndpoint",
        "sagemaker:UpdateEndpointWeightsAndCapacities"
      ],
      "Resource": [
        "arn:aws:sagemaker:*:*:*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:ListEndpointConfigs",
        "sagemaker:ListEndpoints",
        "sagemaker:ListHyperParameterTuningJobs",
        "sagemaker:ListModels",
        "sagemaker:ListProcessingJobs",
        "sagemaker:ListTrainingJobs",
        "sagemaker:ListTransformJobs"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:AbortMultipartUpload",
```

```
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Saat membuat peran ini, edit hubungan kepercayaan sehingga berbunyi sebagai berikut:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "ec2.amazonaws.com",
          "rds.amazonaws.com",
          "sagemaker.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Terakhir, salin ARN yang ditugaskan untuk peran baru `NeptuneSageMakerIAMRole` ini.

Important

- Pastikan izin Amazon S3 `NeptuneSageMakerIAMRole` sesuai dengan yang di atas.
- ARN universal, `arn:aws:s3:::*` digunakan untuk sumber daya Amazon S3 dalam kebijakan di atas. Jika karena alasan tertentu ARN universal tidak dapat digunakan, maka

`arn:aws:s3:::graphlytics*` dan ARN untuk sumber daya Amazon S3 pelanggan lain yang akan digunakan perintah Neptuneml harus ditambahkan ke bagian sumber daya.

Konfigurasikan cluster DB Anda untuk mengaktifkan Neptunus ML

Untuk mengatur cluster DB Anda untuk Neptunus ML

1. Di konsol Neptunus, navigasikan ke Grup Parameter dan kemudian ke grup parameter cluster DB yang terkait dengan cluster DB yang akan Anda gunakan. Atur `neptune_ml_iam_role` parameter ke ARN yang ditetapkan ke `NeptuneSageMakerIAMRole` peran yang baru saja Anda buat.
2. Arahkan ke Database, lalu pilih cluster DB yang akan Anda gunakan untuk Neptunus ML. Pilih Tindakan lalu Kelola peran IAM.
3. Pada halaman Kelola peran IAM, pilih Tambah peran dan tambahkan. `NeptuneSageMakerIAMRole` Kemudian tambahkan `NeptuneLoadFromS3` peran.
4. Reboot instance penulis dari cluster DB Anda.

Buat dua titik akhir SageMaker AI di VPC Neptunus Anda

Terakhir, untuk memberi akses mesin Neptunus APIs manajemen AI yang SageMaker diperlukan, Anda perlu membuat SageMaker dua titik akhir AI di VPC Neptunus Anda, seperti yang dijelaskan dalam. Buat dua titik akhir untuk SageMaker AI di VPC Neptunus Anda

Mengkonfigurasi notebook Neptunus secara manual untuk Neptunus ML

Notebook SageMaker Neptunus AI hadir dengan berbagai notebook sampel untuk Neptunus ML. Anda dapat melihat pratinjau sampel ini di repositori grafik-notebook GitHub sumber terbuka.

Anda dapat menggunakan salah satu notebook Neptunus yang ada, atau jika mau, Anda dapat membuatnya sendiri, mengikuti instruksi di. Menggunakan workbench Neptune untuk meng-host notebook Neptune

Anda juga dapat mengonfigurasi notebook Neptunus default untuk digunakan dengan Neptunus ML dengan mengikuti langkah-langkah berikut:

Memodifikasi notebook untuk Neptunus ML

1. Buka konsol Amazon SageMaker AI di https://console.aws.amazon.com/sagemaker/.

2. Pada panel navigasi di sebelah kiri, pilih Notebook, lalu Instans Notebook. Cari nama notebook Neptunus yang ingin Anda gunakan untuk Neptunus ML dan pilih untuk membuka halaman detailnya.
3. Jika instance notebook sedang berjalan, pilih tombol Stop di kanan atas halaman detail notebook.
4. Di pengaturan instans Notebook, di bawah Konfigurasi Siklus Hidup, pilih tautan untuk membuka halaman siklus hidup buku catatan.
5. Pilih Edit di kanan atas, lalu Lanjutkan.
6. Di tab Start notebook, ubah skrip untuk menyertakan perintah ekspor tambahan dan untuk mengisi bidang untuk peran Neptunus MLIAM Anda dan URI layanan Ekspor, sesuatu seperti ini tergantung pada shell Anda:

```
echo "export NEPTUNE_ML_ROLE_ARN=(your Neptune ML IAM role ARN)" >> ~/.bashrc
echo "export NEPTUNE_EXPORT_API_URI=(your export service URI)" >> ~/.bashrc
```

7. Pilih Perbarui.
8. Kembali ke halaman instance notebook. Di bawah Izin dan enkripsi ada bidang untuk peran IAM ARN. Pilih tautan di bidang ini untuk pergi ke peran IAM yang dijalankan oleh instance notebook ini.
9. Buat kebijakan inline baru seperti ini:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "arn:aws:cloudwatch:us-east-1:111122223333:*",
      "Sid": "AllowPutMetrics",
      "Effect": "Allow"
    },
    {
      "Action": [
        "logs:CreateLogGroup",
```

```
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "arn:aws:logs:us-east-1:111122223333:*",
      "Sid": "AllowCreateLogs",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:Put*",
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "arn:aws:s3:::*",
      "Sid": "AllowS3Actions",
      "Effect": "Allow"
    },
    {
      "Action": "execute-api:Invoke",
      "Resource": "arn:aws:execute-api:us-east-1:111122223333:*/*",
      "Sid": "AllowExecute",
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:CreateModel",
        "sagemaker:CreateEndpointConfig",
        "sagemaker:CreateEndpoint",
        "sagemaker:DescribeModel",
        "sagemaker:DescribeEndpointConfig",
        "sagemaker:DescribeEndpoint",
        "sagemaker:DeleteModel",
        "sagemaker:DeleteEndpointConfig",
        "sagemaker:DeleteEndpoint"
      ],
      "Resource": "arn:aws:sagemaker:us-east-1:111122223333:*/*",
      "Sid": "AllowApiActions",
      "Effect": "Allow"
    },
    {
      "Action": [
        "iam:PassRole"
```

```
      ],
      "Resource": "arn:aws:iam::111122223333:role/role-name",
      "Sid": "AllowPassRole",
      "Effect": "Allow"
    }
  ]
}
```

10. Simpan kebijakan baru ini dan lampirkan ke peran IAM di Langkah 8.
11. Pilih Mulai di kanan atas halaman detail instans notebook SageMaker AI untuk memulai instance notebook.

Menggunakan AWS CLI untuk mengatur Neptunus MLpada cluster DB

Selain template CloudFormation mulai cepat dan Konsol Manajemen AWS, Anda juga dapat mengatur Neptunus ML menggunakan file. AWS CLI

Buat grup parameter cluster DB untuk cluster Neptunus MLmu yang baru

AWS CLI Perintah berikut membuat grup parameter cluster DB baru dan mengaturnya untuk bekerja dengan Neptunus ML:

Untuk membuat dan mengkonfigurasi grup parameter klaster DB untuk Neptune ML

1. Buat grup parameter klaster DB baru:

```
aws neptune create-db-cluster-parameter-group \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --db-parameter-group-family neptune1
  --description "(description of your machine learning project)" \
  --region (AWS region, such as us-east-1)
```

2. Buat parameter cluster `neptune_ml_iam_role` DB yang disetel ke ARN `SageMakerExcecutionIAMRole` untuk digunakan cluster DB Anda saat memanggil SageMaker AI untuk membuat pekerjaan dan mendapatkan prediksi dari model ML-host:

```
aws neptune modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --parameters "ParameterName=neptune_ml_iam_role, \
                ParameterValue=ARN of the SageMakerExcecutionIAMRole, \
                Description=NeptuneMLRole, \
                ApplyMethod=pending-reboot" \
  --region (AWS region, such as us-east-1)
```

   Menyetel parameter ini memungkinkan Neptunus untuk SageMaker mengakses AI tanpa Anda harus meneruskan peran dengan setiap panggilan.

   Untuk informasi tentang cara membuat `SageMakerExcecutionIAMRole`, lihat Buat NeptuneSageMaker IAMRole peran khusus.

3. Akhirnya, gunakan `describe-db-cluster-parameters` untuk memeriksa bahwa semua parameter dalam grup parameter klaster DB baru ditetapkan seperti yang Anda inginkan menjadi:

```
aws neptune describe-db-cluster-parameters \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --region (AWS region, such as us-east-1)
```

Melampirkan grup parameter klaster DB baru untuk klaster DB yang Anda gunakan dengan Neptune ML

Sekarang Anda dapat melampirkan grup parameter klaster DB baru yang baru saja Anda buat untuk klaster DB yang ada dengan menggunakan perintah berikut:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (the name of your existing DB cluster) \
  --apply-immediately
  --db-cluster-parameter-group-name (name of your new DB cluster parameter group) \
  --region (AWS region, such as us-east-1)
```

Untuk membuat semua parameter efektif, Anda dapat me-reboot klaster DB:

```
aws neptune reboot-db-instance
  --db-instance-identifier (name of the primary instance of your DB cluster) \
  --profile (name of your AWS profile to use) \
  --region (AWS region, such as us-east-1)
```

Atau, jika Anda membuat klaster DB baru untuk digunakan dengan Neptune ML, Anda dapat menggunakan perintah berikut untuk membuat klaster dengan grup parameter baru yang dilampirkan, dan kemudian buat instans (penulis) primer baru:

```
cluster-name=(the name of the new DB cluster)
aws neptune create-db-cluster
  --db-cluster-identifier ${cluster-name}
  --engine graphdb \
  --engine-version 1.0.4.1 \
  --db-cluster-parameter-group-name (name of your new DB cluster parameter group) \
  --db-subnet-group-name (name of the subnet to use) \
  --region (AWS region, such as us-east-1)

aws neptune create-db-instance
  --db-cluster-identifier ${cluster-name}
  --db-instance-identifier ${cluster-name}-i \
```

```
  --db-instance-class (the instance class to use, such as db.r5.xlarge)
  --engine graphdb \
  --region (AWS region, such as us-east-1)
```

Lampirkan **`NeptuneSageMakerIAMRole`** ke cluster DB Anda sehingga dapat mengakses sumber daya SageMaker AI dan Amazon S3

Terakhir, ikuti instruksi Buat NeptuneSageMaker IAMRole peran khusus untuk membuat peran IAM yang memungkinkan cluster DB Anda berkomunikasi dengan SageMaker AI dan Amazon S3. Kemudian, gunakan perintah berikut untuk melampirkan `NeptuneSageMakerIAMRole` peran yang Anda buat ke cluster DB Anda:

```
aws neptune add-role-to-db-cluster
  --db-cluster-identifier ${cluster-name}
  --role-arn arn:aws:iam::(the ARN number of the role's ARN):role/NeptuneMLRole \
  --region (AWS region, such as us-east-1)
```

Buat dua titik akhir untuk SageMaker AI di VPC Neptunus Anda

Neptunus L membutuhkan SageMaker dua titik akhir AI di VPC cluster Neptunus DB Anda:

- `com.amazonaws.(AWS region, like us-east-1).sagemaker.runtime`
- `com.amazonaws.(AWS region, like us-east-1).sagemaker.api`

Jika Anda belum menggunakan CloudFormation templat mulai cepat, yang membuatnya secara otomatis untuk Anda, Anda dapat menggunakan AWS CLI perintah berikut untuk membuatnya:

Yang satu ini membuat titik akhir `sagemaker.runtime`:

```
aws ec2 create-vpc-endpoint
  --vpc-id (the ID of your Neptune DB cluster's VPC)
  --vpc-endpoint-type Interface
  --service-name com.amazonaws.(AWS region, like us-east-1).sagemaker.runtime
  --subnet-ids (the subnet ID or IDs that you want to use)
  --security-group-ids (the security group for the endpoint network interface, or omit
 to use the default)
  --private-dns-enabled
```

Dan yang satu ini membuat titik akhir `sagemaker.api`:

```
aws ec2 create-vpc-endpoint
  --vpc-id (the ID of your Neptune DB cluster's VPC)
  --vpc-endpoint-type Interface
  --service-name com.amazonaws.(AWS region, like us-east-1).sagemaker.api
  --subnet-ids (the subnet ID or IDs that you want to use)
  --security-group-ids (the security group for the endpoint network interface, or omit
 to use the default)
  --private-dns-enabled
```

Anda juga dapat menggunakan Konsol VPC untuk membuat titik akhir ini. Lihat Panggilan prediksi aman di Amazon SageMaker dengan AWS PrivateLink dan Mengamankan semua panggilan SageMaker API Amazon dengan. AWS PrivateLink

Buat parameter titik akhir inferensi SageMaker AI di grup parameter cluster DB Anda

Untuk menghindari keharusan menentukan titik akhir inferensi SageMaker AI dari model yang Anda gunakan di setiap kueri yang Anda buat, buat parameter cluster DB yang dinamai `neptune_ml_endpoint` dalam grup parameter cluster DB untuk Neptunus ML. Mengatur parameter ke `id` dari titik akhir instans di pertanyaan.

Anda dapat menggunakan AWS CLI perintah berikut untuk melakukannya:

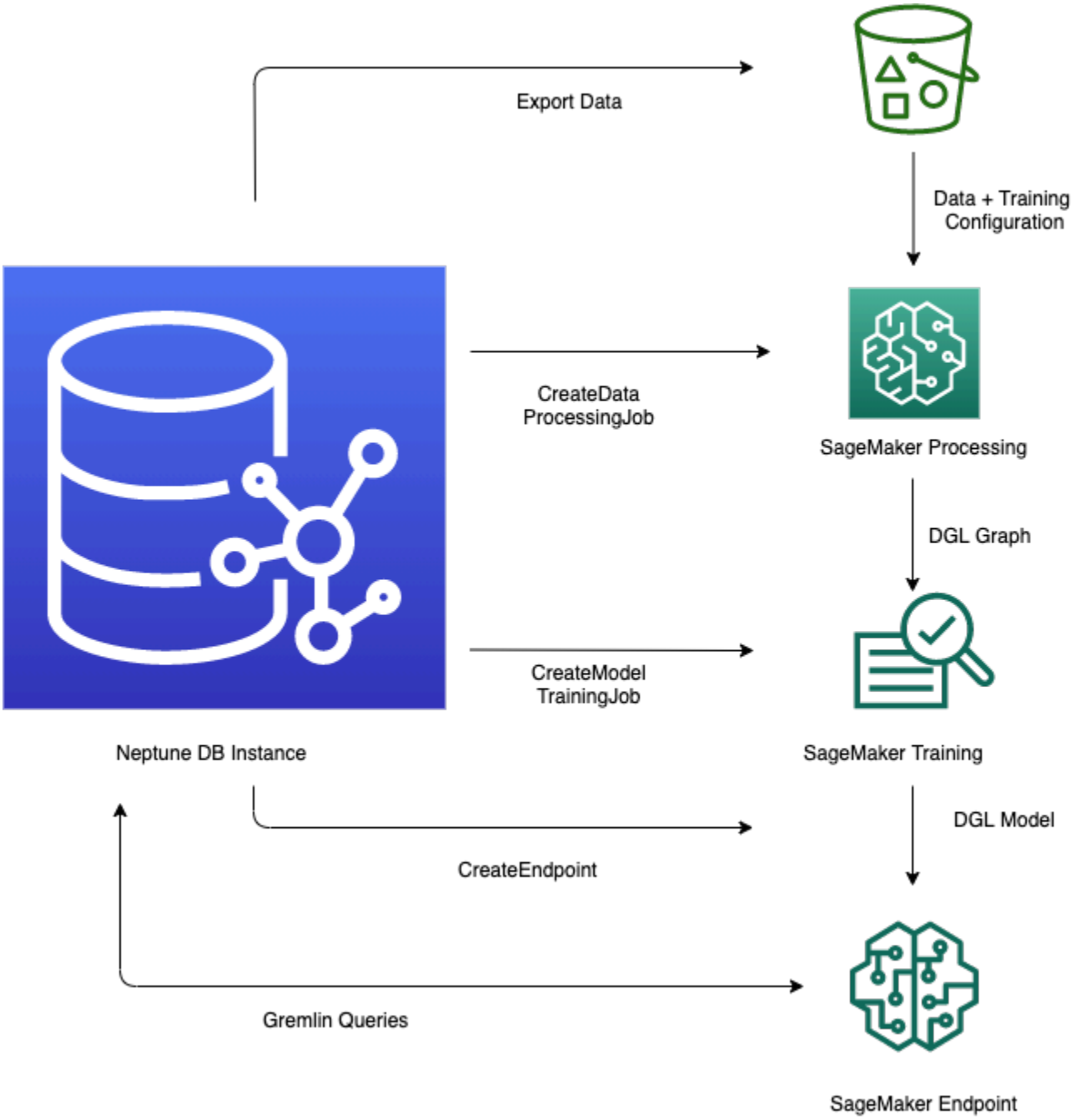
```
aws neptune modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name neptune-ml-demo \
  --parameters "ParameterName=neptune_ml_endpoint, \
                ParameterValue=(the name of the SageMaker AI inference endpoint you
 want to query), \
                Description=NeptuneMLEndpoint, \
                ApplyMethod=pending-reboot" \
  --region (AWS region, such as us-east-1)
```

Gambaran umum cara menggunakan fitur Neptune ML

Fitur Neptunus ML di Amazon Neptunus menyediakan alur kerja yang efisien untuk memanfaatkan model pembelajaran mesin dalam database grafik. Prosesnya melibatkan beberapa langkah kunci - mengekspor data dari Neptunus ke dalam format CSV, memproses data terlebih dahulu untuk mempersiapkannya untuk pelatihan model, melatih model pembelajaran mesin menggunakan SageMaker Amazon AI, membuat titik akhir inferensi untuk menyajikan prediksi, dan kemudian menanyakan model langsung dari kueri Gremlin. Meja kerja Neptunus menyediakan perintah ajaib garis dan sel yang nyaman untuk membantu mengelola dan mengotomatiskan langkah-langkah ini. Dengan mengintegrasikan kemampuan pembelajaran mesin langsung ke dalam database grafik, Neptunus ML memungkinkan pengguna untuk memperoleh wawasan berharga dan membuat prediksi menggunakan data relasional kaya yang disimpan dalam grafik Neptunus.

Memulai alur kerja untuk menggunakan Neptunus ML

Menggunakan fitur Neptunus ML di Amazon Neptunus umumnya melibatkan lima langkah berikut untuk memulai:



1. Ekspor dan konfigurasi data – Langkah ekspor data menggunakan layanan Ekspor Neptune atau alat baris perintah `neptune-export` untuk mengekspor data dari Neptune ke Amazon Simple Storage Service (Amazon S3) dalam bentuk CSV. Sebuah file konfigurasi bernama `training-data-configuration.json` secara otomatis dihasilkan pada saat yang sama, yang menentukan bagaimana data yang diekspor dapat dimuat ke dalam grafik yang dapat dilatih.
2. Prapemrosesan data – Pada langkah ini, set data yang diekspor diproses sebelumnya menggunakan teknik standar guna dipersiapkan untuk pelatihan model. Normalisasi fitur dapat

dilakukan untuk data numerik, dan fitur teks dapat dikodekan menggunakan `word2vec`. Pada akhir langkah ini, grafik DGL (Deep Graph library) dihasilkan dari set data yang diekspor untuk langkah pelatihan model yang akan digunakan.

Langkah ini diterapkan menggunakan pekerjaan pemrosesan SageMaker AI di akun Anda, dan data yang dihasilkan disimpan di lokasi Amazon S3 yang telah Anda tentukan.

3. Pelatihan model – Langkah pelatihan model melatih model pembelajaran mesin yang akan digunakan untuk prediksi.

   Pelatihan model dilakukan dalam dua tahap:

   - Tahap pertama menggunakan pekerjaan pemrosesan SageMaker AI untuk menghasilkan set konfigurasi strategi pelatihan model yang menentukan jenis model dan rentang hiperparameter model apa yang akan digunakan untuk pelatihan model.
   - Tahap kedua kemudian menggunakan pekerjaan penyetelan model SageMaker AI untuk mencoba konfigurasi hyperparameter yang berbeda dan memilih pekerjaan pelatihan yang menghasilkan model berkinerja terbaik. Pekerjaan tuning menjalankan sejumlah uji coba pekerjaan pelatihan model yang telah ditentukan sebelumnya pada data yang diproses. Pada akhir tahap ini, parameter model terlatih dari pekerjaan pelatihan terbaik digunakan untuk menghasilkan artefak model untuk inferensi.

4. Buat titik akhir inferensi di Amazon SageMaker AI — Titik akhir inferensi adalah instance titik akhir SageMaker AI yang diluncurkan dengan artefak model yang dihasilkan oleh pekerjaan pelatihan terbaik. Setiap model terikat pada titik akhir tunggal. Titik akhir mampu menerima permintaan masuk dari basis data grafik dan mengembalikan prediksi model untuk input dalam permintaan. Setelah Anda membuat titik akhir, itu tetap aktif sampai Anda menghapusnya.

5. Query model pembelajaran mesin menggunakan Gremlin – Anda dapat menggunakan ekstensi ke bahasa kueri Gremlin untuk kueri prediksi dari titik akhir inferensi.

> **Note**
>
> Parameter Neptune Workbench berisi magic baris dan magic sel yang dapat menghemat banyak waktu mengelola langkah-langkah ini, yaitu:
>
> - %neptune_ml
> - %%neptune_ml

Membuat prediksi berdasarkan data grafik yang berkembang

Dengan grafik yang terus berubah, Anda mungkin ingin membuat prediksi batch baru secara berkala menggunakan data baru. Menanyakan prediksi yang telah dihitung sebelumnya (inferensi transduktif) dapat secara signifikan lebih cepat daripada menghasilkan prediksi baru dengan cepat berdasarkan data terbaru (inferensi induktif). Kedua pendekatan memiliki tempatnya, tergantung pada seberapa cepat data Anda berubah dan persyaratan kinerja Anda.

Perbedaan antara inferensi induktif dan transduktif

Saat melakukan inferensi transduktif, Neptunus mencari dan mengembalikan prediksi yang telah dihitung sebelumnya pada saat pelatihan.

Saat melakukan inferensi induktif, Neptunus membangun subgraf yang relevan dan mengambil propertinya. Model DGL GNN kemudian menerapkan pemrosesan data dan evaluasi model secara real-time.

Oleh karena itu, inferensi induktif dapat menghasilkan prediksi yang melibatkan node dan tepi yang tidak ada pada saat pelatihan dan yang mencerminkan keadaan grafik saat ini. Ini datang, bagaimanapun, dengan biaya latensi yang lebih tinggi.

Jika grafik Anda dinamis, Anda mungkin ingin menggunakan inferensi induktif untuk memastikan memperhitungkan data terbaru, tetapi jika grafik Anda statis, inferensi transduktif lebih cepat dan lebih efisien.

Inferensi induktif dinonaktifkan secara default. Anda dapat mengaktifkannya untuk kueri dengan menggunakan Neptunus #ml .InductiveInference predikat Gremlin dalam kueri sebagai berikut:

```
.with( "Neptune#ml.inductiveInference")
```

Alur kerja transduktif inkremental

Saat Anda memperbarui artefak model hanya dengan menjalankan kembali langkah satu hingga tiga (dari ekspor Data dan konfigurasi ke transformasi Model), Neptunus MLmendukung cara yang lebih sederhana untuk memperbarui prediksi batch MLmu menggunakan data baru. Salah satunya adalah menggunakan alur kerja model tambahan, dan yang lainnya adalah menggunakan pelatihan ulang model dengan awal yang hangat.

Alur kerja model tambahan

Dalam alur kerja ini, Anda memperbarui prediksi ML tanpa melatih ulang model ML.

> **Note**
>
> Anda hanya dapat melakukan ini ketika data grafik telah diperbarui dengan node dan/atau tepi baru. Saat ini tidak akan berfungsi ketika node dihapus.

1. Ekspor dan konfigurasi data — Langkah ini sama dengan alur kerja utama.
2. Preprocessing data inkremental — Langkah ini mirip dengan langkah preprocessing data dalam alur kerja utama, tetapi menggunakan konfigurasi pemrosesan yang sama yang digunakan sebelumnya, yang sesuai dengan model terlatih tertentu.
3. Transformasi model — Alih-alih langkah pelatihan model, langkah transformasi model ini mengambil model terlatih dari alur kerja utama dan hasil langkah pra-pemrosesan data tambahan, dan menghasilkan artefak model baru untuk digunakan untuk inferensi. Langkah transformasi model meluncurkan pekerjaan pemrosesan SageMaker AI untuk melakukan perhitungan yang menghasilkan artefak model yang diperbarui.
4. Perbarui titik akhir inferensi Amazon SageMaker AI — Secara opsional, jika Anda memiliki titik akhir inferensi yang ada, langkah ini memperbarui titik akhir dengan artefak model baru yang dihasilkan oleh langkah transformasi model. Atau, Anda juga dapat membuat titik akhir inferensi baru dengan artefak model baru.

Model pelatihan ulang dengan awal yang hangat

Dengan menggunakan alur kerja ini, Anda dapat melatih dan menerapkan model ML baru untuk membuat prediksi menggunakan data grafik tambahan, tetapi mulai dari model yang ada yang dihasilkan menggunakan alur kerja utama:

1. Ekspor dan konfigurasi data — Langkah ini sama dengan alur kerja utama.
2. Preprocessing data inkremental — Langkah ini sama dengan alur kerja inferensi model inkremental. Data grafik baru harus diproses dengan metode pemrosesan yang sama yang digunakan sebelumnya untuk pelatihan model.
3. Pelatihan model dengan awal yang hangat — Pelatihan model mirip dengan apa yang terjadi di alur kerja utama, tetapi Anda dapat mempercepat pencarian hiperparameter model dengan memanfaatkan informasi dari tugas pelatihan model sebelumnya.

4. Perbarui titik akhir inferensi Amazon SageMaker AI — Langkah ini sama seperti pada alur kerja inferensi model inkremental.

Alur kerja untuk model kustom di Neptunus ML

Neptunus ML memungkinkan Anda mengimplementasikan, melatih, dan menerapkan model kustom Anda sendiri untuk tugas apa pun yang didukung Neptunus ML. Alur kerja untuk mengembangkan dan menerapkan model khusus pada dasarnya sama dengan model bawaan, dengan beberapa perbedaan, seperti yang dijelaskan dalam. Alur kerja model kustom

Pemilihan instans untuk tahapan Neptunus Neptunus

Tahapan pemrosesan Neptunus ML yang berbeda menggunakan SageMaker instance AI yang berbeda. Di sini, kita membahas bagaimana memilih jenis instance yang tepat untuk setiap tahap. Anda dapat menemukan informasi tentang jenis dan harga instans SageMaker AI di Amazon SageMaker Pricing.

Memilih instance untuk pemrosesan data

Langkah pemrosesan data SageMaker AI memerlukan instance pemrosesan yang memiliki memori dan penyimpanan disk yang cukup untuk data input, perantara, dan output. Jumlah memori dan penyimpanan disk tertentu yang diperlukan tergantung pada karakteristik grafik Neptune ML dan fitur yang diekspor.

Secara default, Neptunus ML memilih instance `ml.r5` terkecil yang memorinya sepuluh kali lebih besar dari ukuran data grafik yang diekspor pada disk.

Memilih contoh untuk pelatihan model dan transformasi model

Memilih jenis instans yang tepat untuk pelatihan model atau transformasi model bergantung pada jenis tugas, ukuran grafik, dan persyaratan putaran Anda. Instans GPU memberikan kinerja terbaik. Kami biasanya merekomendasikan `p3` dan contoh `g4dn` serial. Anda juga dapat menggunakan `p2` atau `p4d` contoh.

Secara default, Neptunus ML memilih instans GPU terkecil dengan lebih banyak memori daripada yang dibutuhkan pelatihan model dan transformasi model. Anda dapat menemukan pilihan apa yang ada di `train_instance_recommendation.json` file, di lokasi keluaran pemrosesan data Amazon S3. Berikut adalah contoh isi `train_instance_recommendation.json` file:

```
{
  "instance":     "(the recommended instance type for model training and transform)",
  "cpu_instance": "(the recommended instance type for base processing instance)",
  "disk_size":    "(the estimated disk space required)",
  "mem_size":     "(the estimated memory required)"
}
```

Memilih instance untuk titik akhir inferensi

Memilih jenis instans yang tepat untuk titik akhir inferensi tergantung pada jenis tugas, ukuran grafik, dan anggaran Anda. Secara default, Neptunus ML memilih instance `ml.m5d` terkecil dengan lebih banyak memori yang dibutuhkan titik akhir inferensi.

> **Note**
>
> Jika lebih dari 384 GB memori diperlukan, Neptunus ML menggunakan sebuah instance. `ml.r5d.24xlarge`

Anda dapat melihat jenis instance apa yang direkomendasikan Neptunus ML dalam file `infer_instance_recommendation.json` yang terletak di lokasi Amazon S3 yang Anda gunakan untuk pelatihan model. Berikut adalah contoh isi file itu:

```
{
  "instance" :   "(the recommended instance type for an inference endpoint)",
  "disk_size" :  "(the estimated disk space required)",
  "mem_size" :   "(the estimated memory required)"
}
```

Menggunakan alat ekspor neptunus atau layanan Ekspor Neptunus untuk mengekspor data dari Neptunus untuk Neptunus

Neptunus ML mengharuskan Anda menyediakan data pelatihan untuk Deep Graph Library (DGL) untuk membuat dan mengevaluasi model.

Anda dapat mengekspor data dari Neptunus menggunakan layanan Neptunus-Ekspor, atau utilitas. `neptune-export` Baik layanan dan alat baris perintah mempublikasikan data ke Amazon Simple Storage Service (Amazon S3) dalam format CSV, dienkripsi menggunakan enkripsi sisi server Amazon S3 (). `SSE-S3` Lihat File yang diekspor oleh Neptune-Ekspor dan `neptune-export`.

Selain itu, saat Anda mengonfigurasi ekspor data pelatihan untuk Neptunus, tugas ekspor akan membuat dan menerbitkan file konfigurasi pelatihan model terenkripsi bersama dengan data yang diekspor. Secara default, file ini diberi nama`training-data-configuration.json`.

Contoh penggunaan layanan Neptunus-Ekspor untuk mengekspor data pelatihan untuk Neptunus ML

Permintaan ini mengekspor data pelatihan grafik properti untuk tugas klasifikasi simpul:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-pg",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": {
          "endpoint": "(your Neptune endpoint DNS name)",
          "profile": "neptune_ml"
        },
        "additionalParams": {
          "neptune_ml": {
            "version": "v2.0",
            "targets": [
              {
                "node": "Movie",
                "property": "genre",
                "type": "classification"
              }
```

```
            ]
          }
        }
      }'
```

Permintaan ini mengekspor data pelatihan RDF untuk tugas klasifikasi node:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-rdf",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": {
          "endpoint": "(your Neptune endpoint DNS name)",
          "profile": "neptune_ml"
        },
        "additionalParams": {
          "neptune_ml": {
            "version": "v2.0",
            "targets": [
              {
                "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
                "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/
genre",
                "type": "classification"
              }
            ]
          }
        }
      }'
```

Bidang yang akan disetel di objek params saat mengekspor data pelatihan

`params`Objek dalam permintaan ekspor dapat berisi berbagai bidang, seperti yang dijelaskan dalam `params`dokumentasi. Yang berikut ini paling relevan untuk mengekspor data pelatihan pembelajaran mesin:

- **`endpoint`**— Gunakan `endpoint` untuk menentukan titik akhir instance Neptunus di cluster DB Anda yang proses ekspor dapat kueri untuk mengekstrak data.

- **`profile`**— `profile` Bidang dalam `params` objek harus diatur ke **`neptune-ml`**.

  Hal ini menyebabkan proses ekspor memformat data yang diekspor dengan tepat untuk pelatihan model Neptunus ML, dalam format CSV untuk data grafik properti atau sebagai N-Triple untuk data RDF. Ini juga menyebabkan `training-data-configuration.json` file dibuat dan ditulis ke lokasi Amazon S3 yang sama dengan data pelatihan yang diekspor.
- **`cloneCluster`**— Jika diatur ke`true`, proses ekspor mengkloning cluster DB Anda, mengekspor dari klon, dan kemudian menghapus klon setelah selesai.
- **`useIamAuth`**— Jika cluster DB Anda mengaktifkan otentikasi IAM, Anda harus menyertakan bidang ini yang disetel ke. `true`

Proses ekspor juga menyediakan beberapa cara untuk memfilter data yang Anda ekspor (lihat contoh ini).

Menggunakan objek additionalParams untuk menyetel ekspor informasi pelatihan model

`additionalParams`Objek berisi bidang yang dapat Anda gunakan untuk menentukan label dan fitur kelas pembelajaran mesin untuk tujuan pelatihan dan memandu pembuatan file konfigurasi data pelatihan.

Proses ekspor tidak dapat secara otomatis menyimpulkan properti node dan edge mana yang harus menjadi label kelas pembelajaran mesin untuk dijadikan contoh untuk tujuan pelatihan. Ini juga tidak dapat secara otomatis menyimpulkan pengkodean fitur terbaik untuk properti numerik, kategoris dan teks, jadi Anda perlu menyediakan petunjuk menggunakan bidang dalam `additionalParams` objek untuk menentukan hal-hal ini, atau untuk mengganti pengkodean default.

Untuk data grafik properti, struktur tingkat atas `additionalParams` dalam permintaan ekspor mungkin terlihat seperti ini:

```
{
  "command": "export-pg",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams": {
      "neptune_ml": {
```

```
        "version": "v2.0",
        "targets": [ (an array of node and edge class label targets) ],
        "features": [ (an array of node feature hints) ]
    }
  }
}
```

Untuk data RDF, struktur tingkat atasnya mungkin terlihat seperti ini:

```
{
  "command": "export-rdf",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams": {
      "neptune_ml": {
        "version": "v2.0",
        "targets": [ (an array of node and edge class label targets) ]
    }
  }
}
```

Anda juga dapat menyediakan beberapa konfigurasi ekspor, menggunakan `jobs` bidang:

```
{
  "command": "export-pg",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams" : {
    "neptune_ml" : {
      "version": "v2.0",
      "jobs": [
        {
          "name" : "(training data configuration name)",
          "targets": [ (an array of node and edge class label targets) ],
          "features": [ (an array of node feature hints) ]
        },
        {
```

```
          "name" : "(another training data configuration name)",
          "targets": [ (an array of node and edge class label targets) ],
          "features": [ (an array of node feature hints) ]
        }
      ]
    }
  }
}
```

Elemen tingkat atas di bidang neptune_mldi AdditionalParams

Elemen versi di neptune_ml

Menentukan versi konfigurasi data pelatihan untuk menghasilkan.

(Opsional), Jenis: string, Default: “v2.0".

Jika Anda menyertakan`version`, atur ke`v2.0`.

Bidang pekerjaan di neptune_ml

Berisi array objek konfigurasi data pelatihan, yang masing-masing mendefinisikan pekerjaan pemrosesan data, dan berisi:

- **`name`**— Nama konfigurasi data pelatihan yang akan dibuat.

  Misalnya, konfigurasi data pelatihan dengan nama “job-number-1" menghasilkan file konfigurasi data pelatihan bernama. `job-number-1.json`
- **`targets`**— Array JSON dari target label kelas node dan edge yang mewakili label kelas pembelajaran mesin untuk tujuan pelatihan. Lihat Bidang target dalam objek neptune_ml.
- **`features`**— Sebuah array JSON dari fitur properti node. Lihat Bidang fitur di neptune_ml.

Bidang target dalam objek neptune_ml

`targets`Bidang dalam konfigurasi ekspor data pelatihan JSON berisi array objek target yang menentukan tugas pelatihan dan dan label kelas pembelajaran mesin untuk melatih tugas ini. Isi objek target bervariasi tergantung pada apakah Anda melatih data grafik properti atau data RDF.

Untuk klasifikasi node grafik properti dan tugas regresi, objek target dalam array dapat terlihat seperti ini:

```
{
  "node": "(node property-graph label)",
  "property": "(property name)",
  "type" : "(used to specify classification or regression)",
  "split_rate": [0.8,0.2,0.0],
  "separator": ","
}
```

Untuk tugas klasifikasi tepi grafik properti, regresi atau prediksi tautan, mereka dapat terlihat seperti ini:

```
{
  "edge": "(edge property-graph label)",
  "property": "(property name)",
  "type" : "(used to specify classification, regression or link_prediction)",
  "split_rate": [0.8,0.2,0.0],
  "separator": ","
}
```

Untuk tugas klasifikasi dan regresi RDF, objek target dalam array dapat terlihat seperti ini:

```
{
  "node": "(node type of an RDF node)",
  "predicate": "(predicate IRI)",
  "type" : "(used to specify classification or regression)",
  "split_rate": [0.8,0.2,0.0]
}
```

Untuk tugas prediksi tautan RDF, objek target dalam array dapat terlihat seperti ini::

```
{
  "subject": "(source node type of an edge)",
```

```
  "predicate": "(relation type of an edge)",
  "object": "(destination node type of an edge)",
  "type" : "link_prediction",
  "split_rate": [0.8,0.2,0.0]
}
```

Objek target dapat berisi bidang-bidang berikut:

Daftar Isi

- Bidang dalam objek target grafik properti
  - Bidang node (vertex) dalam objek target
  - Bidang tepi dalam objek target grafik properti
  - Bidang properti dalam objek target grafik properti
  - Bidang tipe dalam objek target grafik properti
  - Bidang split_rate dalam objek target grafik properti
  - Bidang pemisah dalam objek target grafik properti
- Bidang dalam objek target RDF
  - Bidang node dalam objek target RDF
  - Bidang subjek dalam objek target RDF
  - Bidang predikat dalam objek target RDF
  - Bidang objek dalam objek target RDF
  - Bidang tipe dalam objek target RDF
  - split_rateBidang dalam objek target grafik properti

Bidang dalam objek target grafik properti

Bidang node (vertex) dalam objek target

Label grafik properti dari simpul target (simpul). Objek target harus mengandung `node` elemen atau `edge` elemen, tetapi tidak keduanya.

A `node` dapat mengambil salah satu nilai tunggal, seperti ini:

```
  "node": "Movie"
```

Atau, dalam kasus simpul multi-label, dapat mengambil array nilai, seperti ini:

```
"node": ["Content", "Movie"]
```

Bidang tepi dalam objek target grafik properti

Menentukan tepi target dengan label node awal, labelnya sendiri, dan label end-node nya. Objek target harus mengandung `edge` elemen atau `node` elemen, tetapi tidak keduanya.

Nilai `edge` bidang adalah larik JSON dari tiga string yang mewakili label grafik properti simpul awal, label grafik properti dari tepi itu sendiri, dan label grafik properti simpul akhir, seperti ini:

```
"edge": ["Person_A", "knows", "Person_B"]
```

Jika simpul and/or akhir simpul awal memiliki beberapa label, lampirkan dalam array, seperti ini:

```
"edge": [ ["Admin", Person_A"], "knows", ["Admin", "Person_B"] ]
```

Bidang properti dalam objek target grafik properti

Menentukan properti dari simpul target atau tepi, seperti ini:

```
"property" : "rating"
```

Bidang ini diperlukan, kecuali jika tugas target adalah prediksi tautan.

Bidang tipe dalam objek target grafik properti

Menunjukkan jenis tugas target yang akan dilakukan pada `node` atau`edge`, seperti ini:

```
"type" : "regression"
```

Jenis tugas yang didukung untuk node adalah:

- `classification`
- `regression`

Jenis tugas yang didukung untuk tepi adalah:

- `classification`
- `regression`

- `link_prediction`

Bidang ini wajib diisi.

Bidang split_rate dalam objek target grafik properti

(Opsional) Perkiraan proporsi node atau tepi yang akan digunakan masing-masing tahap pelatihan, validasi, dan pengujian. Proporsi ini diwakili oleh array JSON dari tiga angka antara nol dan satu yang menambahkan hingga satu:

```
"split_rate": [0.7, 0.1, 0.2]
```

Jika Anda tidak menyediakan `split_rate` bidang opsional, nilai estimasi default adalah `[0.9, 0.1, 0.0]` untuk tugas klasifikasi dan regresi, dan `[0.9,0.05, 0.05]` untuk tugas prediksi tautan.

Bidang pemisah dalam objek target grafik properti

(Opsional) Digunakan dengan tugas klasifikasi.

`separator`Bidang menentukan karakter yang digunakan untuk membagi nilai properti target menjadi beberapa nilai kategoris ketika digunakan untuk menyimpan beberapa nilai kategori dalam string. Contoh:

```
"separator": "|"
```

Kehadiran `separator` bidang menunjukkan bahwa tugas tersebut adalah tugas klasifikasi multi-target.

Bidang dalam objek target RDF

Bidang node dalam objek target RDF

Mendefinisikan jenis node node target. Digunakan dengan tugas klasifikasi node atau tugas regresi node. Jenis node dari node dalam RDF didefinisikan oleh:

```
  node_id, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, node_type
```

RDF hanya node dapat mengambil satu nilai, seperti ini:

```
  "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

Bidang subjek dalam objek target RDF

Untuk tugas prediksi tautan, `subject` tentukan jenis simpul sumber dari tepi target.

```
  "subject": "http://aws.amazon.com/neptune/csv2rdf/class/Director"
```

> **Note**
>
> Untuk tugas prediksi tautan, `subject` harus digunakan bersama dengan `predicate` dan`object`. Jika salah satu dari ketiganya tidak disediakan, semua sisi diperlakukan sebagai target pelatihan.

Bidang predikat dalam objek target RDF

Untuk tugas klasifikasi node dan regresi node, `predicate` mendefinisikan data literal apa yang digunakan sebagai fitur node target dari node target.

```
  "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/genre"
```

> **Note**
>
> Jika node target hanya memiliki satu predikat yang mendefinisikan fitur simpul target, `predicate` bidang tersebut dapat dihilangkan.

Untuk tugas prediksi tautan, `predicate` tentukan jenis relasi tepi target:

```
"predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/direct"
```

> **Note**
>
> Untuk tugas prediksi tautan, `predicate` harus digunakan bersama dengan `subject` dan`object`. Jika salah satu dari ketiganya tidak disediakan, semua sisi diperlakukan sebagai target pelatihan.

Bidang objek dalam objek target RDF

Untuk tugas prediksi tautan, `object` tentukan jenis simpul tujuan dari tepi target:

```
  "object": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

> **Note**
>
> Untuk tugas prediksi tautan, `object` harus digunakan bersama dengan `subject` dan`predicate`. Jika salah satu dari ketiganya tidak disediakan, semua sisi diperlakukan sebagai target pelatihan.

Bidang tipe dalam objek target RDF

Menunjukkan jenis tugas target yang akan dilakukan, seperti ini:

```
  "type" : "regression"
```

Jenis tugas yang didukung untuk data RDF adalah:

- `link_prediction`
- `classification`
- `regression`

Bidang ini wajib diisi.

`split_rate`Bidang dalam objek target grafik properti

(Opsional) Perkiraan proporsi node atau tepi yang akan digunakan masing-masing tahap pelatihan, validasi, dan pengujian. Proporsi ini diwakili oleh array JSON dari tiga angka antara nol dan satu yang menambahkan hingga satu:

```
"split_rate": [0.7, 0.1, 0.2]
```

Jika Anda tidak menyediakan `split_rate` bidang opsional, nilai estimasi default adalah`[0.9, 0.1, 0.0]`.

Bidang fitur di neptune_ml

Nilai properti dan literal RDF datang dalam berbagai format dan tipe data. Untuk mencapai kinerja yang baik dalam pembelajaran mesin, penting untuk mengubah nilai-nilai tersebut menjadi pengkodean numerik yang dikenal sebagai fitur.

Neptunus ML melakukan ekstraksi fitur dan pengkodean sebagai bagian dari langkah-langkah ekspor data dan pemrosesan data, seperti yang dijelaskan dalam. Pengkodean fitur di Neptunus ML

Untuk kumpulan data grafik properti, proses ekspor secara otomatis menyimpulkan `auto` fitur untuk properti string dan untuk properti numerik yang berisi nilai kelipatan. Untuk properti numerik yang berisi nilai tunggal, ini menyimpulkan fitur. `numerical` Untuk properti tanggal, ia menyimpulkan `datetime` fitur.

Jika Anda ingin mengganti spesifikasi fitur yang disimpulkan secara otomatis, atau menambahkan spesifikasi bucket numerik, TF-IDF, atau SBERT untuk properti FastText, Anda dapat mengontrol pengkodean fitur menggunakan bidang fitur.

> **Note**
>
> Anda hanya dapat menggunakan `features` bidang untuk mengontrol spesifikasi fitur untuk data grafik properti, bukan untuk data RDF.

Untuk teks bentuk bebas, Neptunus ML dapat menggunakan beberapa model berbeda untuk mengubah urutan token dalam nilai properti string menjadi vektor nilai nyata ukuran tetap:

- `text_fasttext`— Menggunakan pengkodean FastText. Ini adalah pengkodean yang disarankan untuk fitur yang menggunakan satu dan hanya satu dari lima bahasa yang didukung FastText.
- `text_sbert`— Menggunakan model pengkodean Sentence BERT (SBERT). Ini adalah pengkodean yang disarankan untuk teks yang `text_fasttext` tidak mendukung.
- `text_word2vec`— Menggunakan algoritma Word2Vec awalnya diterbitkan oleh Google untuk menyandikan teks. Word2Vec hanya mendukung bahasa Inggris.
- `text_tfidf`— Menggunakan istilah frekuensi—inverse document frequency (TF-IDF) vectorizer untuk pengkodean teks. Pengkodean TF-IDF mendukung fitur statistik yang tidak dilakukan pengkodean lainnya.

`features`Bidang berisi array JSON fitur properti node. Objek dalam array dapat berisi bidang-bidang berikut:

Daftar Isi

- Bidang node dalam fitur
- Bidang tepi dalam fitur
- Bidang properti dalam fitur
- Nilai yang mungkin dari bidang tipe untuk fitur
- Bidang norma
- Bidang bahasa
- Bidang max_length
- Bidang pemisah
- Bidang rentang
- Bidang bucket_cnt
- Bidang slide_window_size
- Bidang imputer
- Bidang max_features
- Bidang min_df
- Bidang ngram_range
- Bidang datetime_parts

Bidang node dalam fitur

nodeBidang menentukan label property-graph dari simpul fitur. Contoh:

```
  "node": "Person"
```

Jika sebuah simpul memiliki beberapa label, gunakan array untuk menampungnya. Contoh:

```
  "node": ["Admin", "Person"]
```

Bidang tepi dalam fitur

`edge`Bidang menentukan jenis tepi tepi fitur. Tipe tepi terdiri dari array yang berisi label grafik properti dari simpul sumber, label grafik properti tepi, dan label grafik properti dari simpul tujuan. Anda harus menyediakan ketiga nilai saat menentukan fitur tepi. Contoh:

```
  "edge": ["User", "reviewed", "Movie"]
```

Jika simpul sumber atau tujuan dari tipe tepi memiliki beberapa label, gunakan array lain untuk menampungnya. Contoh:

```
  "edge": [["Admin", "Person"]. "edited", "Post"]
```

Bidang properti dalam fitur

Gunakan parameter properti untuk menentukan properti dari simpul yang diidentifikasi oleh `node` parameter. Contoh:

```
  "property" : "age"
```

Nilai yang mungkin dari bidang tipe untuk fitur

`type`Parameter menentukan jenis fitur yang didefinisikan. Contoh:

```
  "type": "bucket_numerical"
```

Nilai **`type`** parameter yang mungkin

- **`"auto"`**— Menentukan bahwa Neptunus ML harus secara otomatis mendeteksi jenis properti dan menerapkan pengkodean fitur yang tepat. Sebuah `auto` fitur juga dapat memiliki `separator` bidang opsional.

  Lihat Pengkodean fitur otomatis di Neptunus ML.
- **`"category"`**- Pengkodean fitur ini mewakili nilai properti sebagai salah satu dari sejumlah kategori. Dengan kata lain, fitur ini dapat mengambil satu atau lebih nilai diskrit. Sebuah `category` fitur juga dapat memiliki `separator` bidang opsional.

  Lihat Fitur kategoris di Neptunus ML.

- **`"numerical"`**- Pengkodean fitur ini mewakili nilai properti numerik sebagai angka dalam interval kontinu di mana “lebih besar dari” dan “kurang dari” memiliki makna.

  Sebuah `numerical` fitur juga dapat memiliki opsionalnorm,`imputer`, dan `separator` bidang.

  Lihat Fitur numerik di Neptunus ML.

- **`"bucket_numerical"`**- Pengkodean fitur ini membagi nilai properti numerik menjadi satu set ember atau kategori.

  Misalnya, Anda dapat menyandikan usia orang dalam 4 ember: anak-anak (0-20), dewasa muda (20-40), paruh baya (40-60), dan orang tua (60 ke atas).

  Sebuah `bucket_numerical` fitur membutuhkan `range` dan `bucket_cnt` bidang, dan secara opsional juga dapat menyertakan `imputer` and/or `slide_window_size` bidang.

  Lihat Fitur bucket-numerik di Neptunus ML.

- **`"datetime"`**— Pengkodean fitur ini mewakili nilai properti datetime sebagai larik fitur kategoris ini: tahun, bulan, hari kerja, dan jam.

  Satu atau lebih dari empat kategori ini dapat dihilangkan menggunakan `datetime_parts` parameter.

  Lihat Fitur Datetime di Neptunus ML.

- **`"text_fasttext"`**- Pengkodean fitur ini mengubah nilai properti yang terdiri dari kalimat atau teks bentuk bebas menjadi vektor numerik menggunakan model FastText. Ini mendukung lima bahasa, yaitu Inggris (`en`), Mandarin (`zh`), Hindi (`hi`), Spanyol (`es`), dan Prancis (`fr`). Untuk nilai properti teks dalam salah satu lima bahasa tersebut, `text_fasttext` adalah pengkodean yang disarankan. Namun, tidak dapat menangani kasus di mana kalimat yang sama berisi kata-kata dalam lebih dari satu bahasa.

  Untuk bahasa lain selain yang didukung FastText, gunakan pengkodean. `text_sbert`

  Jika Anda memiliki banyak string teks nilai properti yang lebih panjang dari, katakanlah, 120 token, gunakan `max_length` bidang untuk membatasi jumlah token di setiap string yang `"text_fasttext"` dikodekan.

  Lihat FastText encoding nilai properti teks di Neptunus ML.

- **`"text_sbert"`**— Pengkodean ini mengubah nilai properti teks menjadi vektor numerik menggunakan model Sentence BERT (SBERT). Neptunus mendukung dua metode SBERT,

`text_sbert128` yaitu, yang merupakan default jika Anda hanya menentukan, dan. `text_sbert` `text_sbert512` Perbedaan di antara mereka adalah jumlah maksimum token dalam properti teks yang dikodekan. `text_sbert128`Pengkodean hanya mengkodekan 128 token pertama, sementara `text_sbert512` mengkodekan hingga 512 token. Akibatnya, penggunaan `text_sbert512` dapat membutuhkan lebih banyak waktu pemrosesan daripada`text_sbert128`. Kedua metode lebih lambat dari`text_fasttext`.

`text_sbert`*Metode ini mendukung banyak bahasa, dan dapat menyandikan kalimat yang berisi lebih dari satu bahasa.

Lihat Sentence BERT (SBERT) encoding fitur teks di Neptunus ML.

- **`"text_word2vec"`**— Pengkodean ini mengubah nilai properti teks menjadi vektor numerik menggunakan algoritma Word2Vec. Ini hanya mendukung bahasa Inggris.

  Lihat Pengkodean Word2Vec fitur teks di Neptunus ML.

- **`"text_tfidf"`**— Pengkodean ini mengubah nilai properti teks menjadi vektor numerik menggunakan istilah frekuensi-inverse document frequency (TF-IDF) vectorizer.

  Anda menentukan parameter pengkodean `text_tfidf` fitur menggunakan `ngram_range` bidang, `min_df` bidang, dan `max_features` bidang.

  Lihat Pengkodean fitur teks TF-IDF di Neptunus.

- **`"none"`**— Menggunakan `none` tipe menyebabkan tidak ada pengkodean fitur yang terjadi. Nilai properti mentah diurai dan disimpan sebagai gantinya.

  Gunakan `none` hanya jika Anda berencana untuk melakukan pengkodean fitur kustom Anda sendiri sebagai bagian dari pelatihan model kustom.

Bidang norma

Bidang ini diperlukan untuk fitur numerik. Ini menentukan metode normalisasi untuk digunakan pada nilai-nilai numerik:

```
"norm": "min-max"
```

Metode normalisasi berikut didukung:

- “min-max” — Normalisasi setiap nilai dengan mengurangi nilai minimum darinya dan kemudian membaginya dengan selisih antara nilai maksimum dan minimum.
- “standar” — Menormalkan setiap nilai dengan membaginya dengan jumlah semua nilai.
- “tidak ada” — Jangan menormalkan nilai numerik selama pengkodean.

Lihat Fitur numerik di Neptunus ML.

Bidang bahasa

Bidang bahasa menentukan bahasa yang digunakan dalam nilai properti teks. Penggunaannya tergantung pada metode pengkodean teks:

- Untuk `text_fasttext`pengkodean, bidang ini diperlukan, dan harus menentukan salah satu bahasa berikut:
  - `en`(Bahasa Inggris)
  - `zh`(Tionghoa)
  - `hi`(Hindi)
  - `es`(Spanyol)
  - `fr`(Perancis)
- Untuk `text_sbert`pengkodean, bidang ini tidak digunakan, karena pengkodean SBERT bersifat multibahasa.
- Untuk `text_word2vec`pengkodean, bidang ini opsional, karena `text_word2vec` hanya mendukung bahasa Inggris. Jika ada, itu harus menentukan nama model bahasa Inggris:

  ```
  "language" : "en_core_web_lg"
  ```

- Untuk `text_tfidf`pengkodean, bidang ini tidak digunakan.

Bidang max_length

`max_length`Bidang ini opsional untuk `text_fasttext` fitur, di mana ia menentukan jumlah maksimum token dalam fitur teks input yang akan dikodekan. Input teks yang lebih panjang dari `max_length` yang terpotong. Misalnya, menyetel max_length ke 128 menunjukkan bahwa token apa pun setelah ke-128 dalam urutan teks akan diabaikan:

```
"max_length": 128
```

Bidang pemisah

Bidang ini digunakan secara opsional dengan`category`, `numerical` dan `auto` fitur. Ini menentukan karakter yang dapat digunakan untuk membagi nilai properti menjadi beberapa nilai kategoris atau nilai numerik:

```
"separator": ";"
```

Hanya gunakan `separator` bidang ketika properti menyimpan beberapa nilai yang dibatasi dalam satu string, seperti `"Actor;Director"` atau. `"0.1;0.2"`

LihatFitur kategoris,Fitur numerik, danPengkodean otomatis.

Bidang rentang

Bidang ini diperlukan untuk `bucket_numerical` fitur. Ini menentukan kisaran nilai numerik yang akan dibagi menjadi ember, dalam format: `[`*`lower-bound`*`, `*`upper-bound`*`]`

```
"range" : [20, 100]
```

Jika nilai properti lebih kecil dari batas bawah maka nilai tersebut ditetapkan ke bucket pertama, atau jika lebih besar dari batas atas, itu ditetapkan ke bucket terakhir.

Lihat Fitur bucket-numerik di Neptunus ML.

Bidang bucket_cnt

Bidang ini diperlukan untuk `bucket_numerical` fitur. Ini menentukan jumlah ember yang rentang numerik yang ditentukan oleh `range` parameter harus dibagi menjadi:

```
"bucket_cnt": 10
```

Lihat Fitur bucket-numerik di Neptunus ML.

Bidang slide_window_size

Bidang ini digunakan secara opsional dengan `bucket_numerical` fitur untuk menetapkan nilai ke lebih dari satu bucket:

```
"slide_window_size": 5
```

Cara kerja jendela slide adalah bahwa Neptunus ML mengambil **s**ukuran jendela dan mengubah setiap **v**nilai numerik properti menjadi rentang dari hingga. `v - s/2` `v + s/2` Nilai tersebut kemudian ditetapkan ke setiap bucket yang rentangnya tumpang tindih.

Lihat Fitur bucket-numerik di Neptunus ML.

Bidang imputer

Bidang ini digunakan secara opsional dengan `numerical` dan `bucket_numerical` fitur untuk memberikan teknik imputasi untuk mengisi nilai yang hilang:

```
"imputer": "mean"
```

Teknik imputasi yang didukung adalah:

- `"mean"`
- `"median"`
- `"most-frequent"`

Jika Anda tidak menyertakan parameter imputer, pra-pemrosesan data akan berhenti dan keluar saat nilai yang hilang ditemukan.

Lihat Fitur numerik di Neptunus ML dan Fitur bucket-numerik di Neptunus ML.

Bidang max_features

Bidang ini digunakan secara opsional oleh `text_tfidf` fitur untuk menentukan jumlah maksimum istilah untuk dikodekan:

```
"max_features": 100
```

Pengaturan 100 menyebabkan vectorizer TF-IDF hanya menyandikan 100 istilah yang paling umum. Nilai default jika Anda tidak menyertakan `max_features` adalah 5.000.

Lihat Pengkodean fitur teks TF-IDF di Neptunus.

Bidang min_df

Bidang ini digunakan secara opsional oleh `text_tfidf` fitur untuk menentukan frekuensi dokumen minimum istilah untuk dikodekan:

```
"min_df": 5
```

Pengaturan 5 menunjukkan bahwa istilah harus muncul dalam setidaknya 5 nilai properti yang berbeda untuk dikodekan.

Nilai default jika Anda tidak menyertakan `min_df` parameter adalah2.

Lihat Pengkodean fitur teks TF-IDF di Neptunus.

Bidang ngram_range

Bidang ini digunakan secara opsional oleh `text_tfidf` fitur untuk menentukan urutan ukuran kata atau token apa yang harus dianggap sebagai istilah individual potensial untuk dikodekan:

```
"ngram_range": [2, 4]
```

Nilai `[2, 4]` menentukan bahwa urutan 2, 3 dan 4 kata harus dianggap sebagai istilah individu potensial.

Default jika Anda tidak secara eksplisit disetel `ngram_range` adalah`[1, 1]`, artinya hanya satu kata atau token yang dianggap sebagai istilah untuk dikodekan.

Lihat Pengkodean fitur teks TF-IDF di Neptunus.

Bidang datetime_parts

Bidang ini digunakan secara opsional oleh `datetime` fitur untuk menentukan bagian mana dari nilai datetime yang akan dikodekan secara kategoris:

```
"datetime_parts": ["weekday", "hour"]
```

Jika Anda tidak menyertakan`datetime_parts`, secara default Neptunus ML mengkodekan bagian tahun, bulan, hari kerja, dan jam dari nilai datetime. Nilai `["weekday", "hour"]` menunjukkan bahwa hanya hari kerja dan jam nilai datetime yang harus dikodekan secara kategoris dalam fitur.

Jika salah satu bagian tidak memiliki lebih dari satu nilai unik dalam set pelatihan, itu tidak dikodekan.

Lihat Fitur Datetime di Neptunus ML.

Contoh penggunaan parameter dalam AdditionalParams untuk menyetel konfigurasi pelatihan model

Contoh berikut menunjukkan bagaimana memanfaatkan fitur “AdditionalParams” dalam grafik properti dan model data RDF untuk mengonfigurasi berbagai aspek proses pelatihan model untuk aplikasi Neptunus ML. Contohnya mencakup berbagai fungsi, termasuk menentukan tingkat pemisahan default untuk training/validation/test data, mendefinisikan klasifikasi node, regresi, dan tugas prediksi tautan, serta mengonfigurasi berbagai jenis fitur seperti bucket numerik, penyematan teks, datetime, dan data kategoris. Konfigurasi terperinci ini memungkinkan Anda untuk menyesuaikan pipeline pembelajaran mesin dengan data spesifik dan persyaratan pemodelan Anda, membuka potensi penuh dari kemampuan Neptunus ML.

Daftar Isi

- Contoh grafik properti menggunakan AdditionalParams
  - Menentukan tingkat pemisahan default untuk konfigurasi pelatihan model
  - Menentukan tugas klasifikasi simpul untuk konfigurasi pelatihan model
  - Menentukan tugas klasifikasi node multi-kelas untuk konfigurasi pelatihan model
  - Menentukan tugas regresi simpul untuk konfigurasi pelatihan model
  - Menentukan tugas klasifikasi tepi untuk konfigurasi pelatihan model
  - Menentukan tugas klasifikasi tepi multi-kelas untuk konfigurasi pelatihan model
  - Menentukan regresi edge untuk konfigurasi model-training
  - Menentukan tugas prediksi tautan untuk konfigurasi pelatihan model
  - Menentukan fitur bucket numerik
  - Menentukan fitur Word2Vec
  - Menentukan fitur FastText
  - Menentukan fitur Sentence BERT
  - Menentukan fitur TF-IDF
  - Menentukan fitur datetime
  - Menentukan fitur category
  - Menentukan fitur numerical
  - Menentukan fitur auto
- Contoh RDF menggunakan additionalParams
  - Menentukan tingkat pemisahan default untuk konfigurasi pelatihan model

- Menentukan tugas klasifikasi simpul untuk konfigurasi pelatihan model
- Menentukan tugas regresi simpul untuk konfigurasi pelatihan model
- Menentukan tugas prediksi tautan untuk tepi tertentu
- Menentukan tugas prediksi tautan untuk semua sisi

Contoh grafik properti menggunakan AdditionalParams

Menentukan tingkat pemisahan default untuk konfigurasi pelatihan model

Dalam contoh berikut, `split_rate` parameter menetapkan tingkat pemisahan default untuk pelatihan model. Jika tidak ada tingkat pemisahan default yang ditentukan, pelatihan menggunakan nilai [0,9, 0,1, 0,0]. Anda dapat mengganti nilai default pada basis per-target dengan menentukan untuk setiap target. `split_rate`

Dalam contoh berikut, `default split_rate` bidang menunjukkan bahwa tingkat perpecahan `[0.7,0.1,0.2]` harus digunakan kecuali diganti berdasarkan per target:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
    "targets": [
      (...)
    ],
    "features": [
      (...)
    ]
  }
}
```

Menentukan tugas klasifikasi simpul untuk konfigurasi pelatihan model

Untuk menunjukkan properti node mana yang berisi contoh berlabel untuk tujuan pelatihan, tambahkan elemen klasifikasi node ke `targets` array, menggunakan`"type" : "classification"`. Tambahkan `split_rate` bidang jika Anda ingin mengganti tingkat pemisahan default.

Dalam contoh berikut, node target menunjukkan bahwa `genre` properti dari setiap `Movie` node harus diperlakukan sebagai label kelas node. `split_rate`Nilai mengesampingkan tingkat pemisahan default:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "genre",
        "type": "classification",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Menentukan tugas klasifikasi node multi-kelas untuk konfigurasi pelatihan model

Untuk menunjukkan properti node mana yang berisi beberapa contoh berlabel untuk tujuan pelatihan, tambahkan elemen klasifikasi node ke array target, gunakan`"type" : "classification"`, dan `separator` untuk menentukan karakter yang dapat digunakan untuk membagi nilai properti target menjadi beberapa nilai kategoris. Tambahkan `split_rate` bidang jika Anda ingin mengganti tingkat pemisahan default.

Dalam contoh berikut, `node` target menunjukkan bahwa `genre` properti dari setiap `Movie` node harus diperlakukan sebagai label kelas node. `separator`Bidang menunjukkan bahwa setiap properti genre berisi beberapa nilai yang dipisahkan titik koma:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "genre",
        "type": "classification",
        "separator": ";"
      }
    ],
    "features": [
      (...)
```

```
    ]
  }
}
```

Menentukan tugas regresi simpul untuk konfigurasi pelatihan model

Untuk menunjukkan properti node mana yang berisi regresi berlabel untuk tujuan pelatihan, tambahkan elemen regresi node ke array target, menggunakan. `"type" : "regression"` Tambahkan bidang split_rate jika Anda ingin mengganti tingkat pemisahan default.

Target `node` berikut in imenunjukkan bahwa properti `rating` masing-masing simpul `Movie` harus diperlakukan sebagai label regresi simpul :

```
    "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "rating",
        "type" : "regression",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
    "features": [
      ...
    ]
  }
}
```

Menentukan tugas klasifikasi tepi untuk konfigurasi pelatihan model

Untuk menunjukkan properti edge mana yang berisi contoh berlabel untuk tujuan pelatihan, tambahkan elemen tepi ke `targets` array, menggunakan`"type" : "regression"`. Tambahkan bidang split_rate jika Anda ingin mengganti tingkat pemisahan default.

`edge`Target berikut menunjukkan bahwa `metAtLocation` properti setiap `knows` tepi harus diperlakukan sebagai label kelas tepi:

```
  "additionalParams": {
  "neptune_ml": {
```

```
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "knows", "Person"],
        "property": "metAtLocation",
        "type": "classification"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Menentukan tugas klasifikasi tepi multi-kelas untuk konfigurasi pelatihan model

Untuk menunjukkan properti edge mana yang berisi beberapa contoh berlabel untuk tujuan pelatihan, tambahkan elemen tepi ke `targets` array, menggunakan`"type" : "classification"`, dan `separator` bidang untuk menentukan karakter yang digunakan untuk membagi nilai properti target menjadi beberapa nilai kategoris. Tambahkan `split_rate` bidang jika Anda ingin mengganti tingkat pemisahan default.

edgeTarget berikut menunjukkan bahwa `sentiment` properti setiap `repliedTo` tepi harus diperlakukan sebagai label kelas tepi. Bidang pemisah menunjukkan bahwa setiap properti sentimen berisi nilai yang dipisahkan koma multile:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "repliedTo", "Message"],
        "property": "sentiment",
        "type": "classification",
        "separator": ","
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Menentukan regresi edge untuk konfigurasi model-training

Untuk menunjukkan properti edge mana yang berisi contoh regresi berlabel untuk tujuan pelatihan, tambahkan `edge` elemen ke `targets` array, menggunakan. `"type" : "regression"` Tambahkan `split_rate` bidang jika Anda ingin mengganti tingkat pemisahan default.

edgeTarget berikut menunjukkan bahwa `rating` properti setiap `reviewed` tepi harus diperlakukan sebagai regresi tepi:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "reviewed", "Movie"],
        "property": "rating",
        "type" : "regression"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Menentukan tugas prediksi tautan untuk konfigurasi pelatihan model

Untuk menunjukkan tepi mana yang harus digunakan untuk tujuan pelatihan prediksi tautan, tambahkan elemen tepi ke larik target menggunakan`"type" : "link_prediction"`. Tambahkan `split_rate` bidang jika Anda ingin mengganti tingkat pemisahan default.

edgeTarget berikut menunjukkan bahwa `cites` tepi harus digunakan untuk prediksi tautan:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Article", "cites", "Article"],
        "type" : "link_prediction"
      }
    ],
    "features": [
```

```
      (...)
    ]
  }
}
```

Menentukan fitur bucket numerik

Anda dapat menentukan fitur data numerik untuk properti node dengan menambahkan `"type": "bucket_numerical"` ke `features` array.

Fitur node berikut ini menunjukkan bahwa properti `age` masing-masing node `Person` harus diperlakukan sebagai fitur bucket numerik:

```
  "additionalParams": {
  "neptune_ml": {
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Person",
        "property": "age",
        "type": "bucket_numerical",
        "range": [1, 100],
        "bucket_cnt": 5,
        "slide_window_size": 3,
        "imputer": "median"
      }
    ]
  }
}
```

Menentukan fitur **`Word2Vec`**

Anda dapat menentukan `Word2Vec` fitur untuk properti node dengan menambahkan `"type": "text_word2vec"` ke `features` array.

Fitur node berikut ini menunjukkan bahwa properti `description` masing-masing node `Movie`harus diperlakukan sebagai fitur `Word2Vec`:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
```

```
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_word2vec",
        "language": "en_core_web_lg"
      }
    ]
  }
}
```

Menentukan fitur **`FastText`**

Anda dapat menentukan `FastText` fitur untuk properti node dengan menambahkan `"type": "text_fasttext"` ke `features` array. `language`Bidang diperlukan, dan harus menentukan salah satu kode bahasa berikut:

- `en`(Bahasa Inggris)
- `zh`(Tionghoa)
- `hi`(Hindi)
- `es`(Spanyol)
- `fr`(Perancis)

Perhatikan bahwa `text_fasttext` pengkodean tidak dapat menangani lebih dari satu bahasa pada satu waktu dalam suatu fitur.

nodeFitur berikut menunjukkan bahwa `description` properti Prancis dari setiap `Movie` node harus diperlakukan sebagai `FastText` fitur:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
```

```
        "node": "Movie",
        "property": "description",
        "type": "text_fasttext",
        "language": "fr",
        "max_length": 1024
      }
    ]
  }
}
```

Menentukan fitur **`Sentence BERT`**

Anda dapat menentukan `Sentence BERT` fitur untuk properti node dengan menambahkan `"type": "text_sbert"` ke `features` array. Anda tidak perlu menentukan bahasa, karena metode ini secara otomatis mengkodekan fitur teks menggunakan model bahasa multibahasa.

Fitur node berikut ini menunjukkan bahwa properti `description` masing-masing node `Movie`harus diperlakukan sebagai fitur `Sentence BERT`:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_sbert128",
      }
    ]
  }
}
```

Menentukan fitur **`TF-IDF`**

Anda dapat menentukan `TF-IDF` fitur untuk properti node dengan menambahkan `"type": "text_tfidf"` ke `features` array.

Fitur node berikut ini menunjukkan bahwa properti `bio` masing-masing node `Person`harus diperlakukan sebagai fitur `TF-IDF`:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "bio",
        "type": "text_tfidf",
        "ngram_range": [1, 2],
        "min_df": 5,
        "max_features": 1000
      }
    ]
  }
}
```

Menentukan fitur **`datetime`**

Proses ekspor secara otomatis menyimpulkan `datetime` fitur untuk properti tanggal. Namun, jika Anda ingin membatasi `datetime_parts` penggunaan untuk `datetime` fitur, atau mengganti spesifikasi fitur sehingga properti yang biasanya diperlakukan sebagai fitur secara eksplisit diperlakukan sebagai `auto` `datetime` fitur, Anda dapat melakukannya dengan menambahkan a `"type": "datetime"` ke array fitur.

Fitur `node` berikut ini menunjukkan bahwa properti `createdAt` masing-masing node `Post`harus diperlakukan sebagai fitur `datetime`:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Post",
        "property": "createdAt",
        "type": "datetime",
        "datetime_parts": ["month", "weekday", "hour"]
```

```
      }
    ]
  }
}
```

Menentukan fitur **`category`**

Proses ekspor secara otomatis menyimpulkan `auto` fitur untuk properti string dan properti numerik yang berisi nilai kelipatan. Untuk properti numerik yang berisi nilai tunggal, ini menyimpulkan fitur. `numerical` Untuk properti tanggal, ia menyimpulkan `datetime` fitur.

Jika Anda ingin mengganti spesifikasi fitur sehingga properti diperlakukan sebagai fitur kategoris, tambahkan `"type": "category"` ke array fitur. Jika properti berisi beberapa nilai, sertakan `separator` bidang. Contoh:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Post",
        "property": "tag",
        "type": "category",
        "separator": "|"
      }
    ]
  }
}
```

Menentukan fitur **`numerical`**

Proses ekspor secara otomatis menyimpulkan `auto` fitur untuk properti string dan properti numerik yang berisi nilai kelipatan. Untuk properti numerik yang berisi nilai tunggal, ini menyimpulkan fitur. `numerical` Untuk properti tanggal, ia menyimpulkan `datetime` fitur.

Jika Anda ingin mengganti spesifikasi fitur sehingga properti diperlakukan sebagai `numerical` fitur, tambahkan `"type": "numerical"` ke array fitur. Jika properti berisi beberapa nilai, sertakan `separator` bidang. Contoh:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Recording",
        "property": "duration",
        "type": "numerical",
        "separator": ","
      }
    ]
  }
}
```

Menentukan fitur **`auto`**

Proses ekspor secara otomatis menyimpulkan `auto` fitur untuk properti string dan properti numerik yang berisi nilai kelipatan. Untuk properti numerik yang berisi nilai tunggal, ini menyimpulkan fitur. `numerical` Untuk properti tanggal, ia menyimpulkan `datetime` fitur.

Jika Anda ingin mengganti spesifikasi fitur sehingga properti diperlakukan sebagai `auto` fitur, tambahkan `"type": "auto"` ke array fitur. Jika properti berisi beberapa nilai, sertakan `separator` bidang. Contoh:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "User",
        "property": "role",
        "type": "auto",
        "separator": ","
      }
    ]
  }
```

```
}
```

Contoh RDF menggunakan **additionalParams**

Menentukan tingkat pemisahan default untuk konfigurasi pelatihan model

Dalam contoh berikut, `split_rate` parameter menetapkan tingkat pemisahan default untuk pelatihan model. Jika tidak ada tingkat pemisahan default yang ditentukan, pelatihan menggunakan nilai [0,9, 0,1, 0,0]. Anda dapat mengganti nilai default pada basis per-target dengan menentukan untuk setiap target. `split_rate`

Dalam contoh berikut, `default split_rate` bidang menunjukkan bahwa tingkat perpecahan `[0.7,0.1,0.2]` harus digunakan kecuali diganti berdasarkan per target:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
    "targets": [
      (...)
    ]
  }
}
```

Menentukan tugas klasifikasi simpul untuk konfigurasi pelatihan model

Untuk menunjukkan properti node mana yang berisi contoh berlabel untuk tujuan pelatihan, tambahkan elemen klasifikasi node ke `targets` array, menggunakan`"type" : "classification"`. Tambahkan bidang node untuk menunjukkan jenis node dari node target. Tambahkan `predicate` bidang untuk menentukan data literal mana yang digunakan sebagai fitur simpul target dari simpul target. Tambahkan `split_rate` bidang jika Anda ingin mengganti tingkat pemisahan default.

Dalam contoh berikut, node target menunjukkan bahwa `genre` properti dari setiap `Movie` node harus diperlakukan sebagai label kelas node. `split_rate`Nilai mengesampingkan tingkat pemisahan default:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
```

```
      {
        "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/genre",
        "type": "classification",
        "split_rate": [0.7,0.1,0.2]
      }
    ]
  }
}
```

Menentukan tugas regresi simpul untuk konfigurasi pelatihan model

Untuk menunjukkan properti node mana yang berisi regresi berlabel untuk tujuan pelatihan, tambahkan elemen regresi node ke array target, menggunakan. `"type" : "regression"` Tambahkan `node` bidang untuk menunjukkan jenis node dari node target. Tambahkan `predicate` bidang untuk menentukan data literal mana yang digunakan sebagai fitur simpul target dari simpul target. Tambahkan `split_rate` bidang jika Anda ingin mengganti tingkat pemisahan default.

Target `node` berikut in imenunjukkan bahwa properti `rating` masing-masing simpul `Movie` harus diperlakukan sebagai label regresi simpul :

```
    "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/rating",
        "type": "regression",
        "split_rate": [0.7,0.1,0.2]
      }
    ]
  }
}
```

Menentukan tugas prediksi tautan untuk tepi tertentu

Untuk menunjukkan tepi mana yang harus digunakan untuk tujuan pelatihan prediksi tautan, tambahkan elemen tepi ke larik target menggunakan`"type" : "link_prediction"`. Tambahkan`subject`, `predicate` dan `object` bidang untuk menentukan jenis tepi. Tambahkan `split_rate` bidang jika Anda ingin mengganti tingkat pemisahan default.

edgeTarget berikut menunjukkan bahwa `directed` tepi yang `Directors` terhubung `Movies` harus digunakan untuk prediksi tautan:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "subject": "http://aws.amazon.com/neptune/csv2rdf/class/Director",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/directed",
        "object": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "type" : "link_prediction"
      }
    ]
  }
}
```

Menentukan tugas prediksi tautan untuk semua sisi

Untuk menunjukkan bahwa semua tepi harus digunakan untuk tujuan pelatihan prediksi tautan, tambahkan `edge` elemen ke array target menggunakan`"type" : "link_prediction"`. Jangan menambahkan`subject`,`predicate`, atau `object` bidang. Tambahkan `split_rate` bidang jika Anda ingin mengganti tingkat pemisahan default.

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "type" : "link_prediction"
      }
    ]
  }
}
```

Memproses data grafik yang diekspor dari Neptune untuk pelatihan

Langkah pemrosesan data mengambil data grafik Neptune yang dibuat oleh proses ekspor dan menciptakan informasi yang digunakan oleh Deep Graph Library (DGL) selama pelatihan. Ini termasuk melakukan berbagai pemetaan data dan transformasi:

- Simpul parsing dan edge untuk membangun grafik- dan file pemetaan ID yang dibutuhkan oleh DGL.
- Mengkonversi properti simpul dan fitur simpul dan edge yang dibutuhkan oleh DGL.
- Memisahkan data ke dalam pelatihan, validasi, dan set tes.

Mengelola langkah pemrosesan data untuk Neptune ML

Setelah Anda mengekspor data dari Neptune yang ingin Anda gunakan untuk pelatihan model, Anda dapat memulai tugas pemrosesan data menggunakan perintah `curl` (atau`awscurl`) seperti berikut ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)",
        "configFileName" : "training-job-configuration.json"
      }'
```

Rincian tentang cara menggunakan perintah ini dijelaskan dalam Perintah pemrosesandata, bersama dengan informasi tentang cara untuk mendapatkan status tugas yang sedang berjalan, cara menghentikan tugas yang sedang berjalan, dan mendaftar semua pekerjaan yang sedang berjalan.

Memproses data grafik yang diperbarui untuk Neptunus ML

Anda juga dapat menyediakan a `previousDataProcessingJobId` ke API untuk memastikan bahwa pekerjaan pemrosesan data baru menggunakan metode pemrosesan yang sama dengan pekerjaan sebelumnya. Ini diperlukan saat Anda ingin mendapatkan prediksi untuk data grafik

yang diperbarui di Neptunus, baik dengan melatih ulang model lama pada data baru, atau dengan menghitung ulang artefak model pada data baru.

Anda melakukan ini dengan menggunakan perintah `curl` (atau`awscurl`) seperti ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{ "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
       "id" : "(a job ID for the new job)",
       "processedDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your output
 folder)",
       "previousDataProcessingJobId", "(the job ID of the previous data-processing
 job)"}'
```

Tetapkan nilai `previousDataProcessingJobId` parameter ke ID pekerjaan dari pekerjaan pemrosesan data sebelumnya yang sesuai dengan model terlatih.

> **Note**
>
> Penghapusan node dalam grafik yang diperbarui saat ini tidak didukung. Jika node telah dihapus dalam grafik yang diperbarui, Anda harus memulai pekerjaan pemrosesan data yang sama sekali baru daripada menggunakan`previousDataProcessingJobId`.

Pengkodean fitur di Neptunus ML

Nilai properti datang dalam berbagai format dan tipe data. Untuk mencapai kinerja yang baik dalam pembelajaran mesin, penting untuk mengubah nilai-nilai tersebut menjadi pengkodean numerik yang dikenal sebagai fitur.

Neptunus ML melakukan ekstraksi fitur dan pengkodean sebagai bagian dari langkah-langkah ekspor data dan pemrosesan data, menggunakan teknik pengkodean fitur yang dijelaskan di sini.

> **Note**
>
> Jika Anda berencana untuk mengimplementasikan pengkodean fitur Anda sendiri dalam implementasi model khusus, Anda dapat menonaktifkan pengkodean fitur otomatis dalam tahap pra-pemrosesan data dengan memilih `none` sebagai jenis pengkodean fitur. Tidak ada pengkodean fitur yang terjadi pada properti node atau edge itu, dan sebagai gantinya nilai properti mentah diurai dan disimpan dalam kamus. Preprocessing data masih membuat grafik DGL dari dataset yang diekspor, tetapi grafik DGL yang dibangun tidak memiliki fitur pra-proses untuk pelatihan.
>
> Anda harus menggunakan opsi ini hanya jika Anda berencana untuk melakukan pengkodean fitur kustom Anda sebagai bagian dari pelatihan model khusus. Lihat Model khusus di Neptunus ML untuk detail.

Fitur kategoris di Neptunus ML

Properti yang dapat mengambil satu atau lebih nilai berbeda dari daftar tetap nilai yang mungkin adalah fitur kategoris. Di Neptunus ML, fitur kategoris dikodekan menggunakan pengkodean satu panas. Contoh berikut menunjukkan bagaimana nama properti makanan yang berbeda dikodekan satu panas sesuai dengan kategorinya:

```
 Food        Veg.   Meat   Fruit    Encoding
---------    ----   ----   -----    --------
 Apple         0      0      1         001
 Chicken       0      1      0         010
 Broccoli      1      0      0         100
```

> **Note**
>
> Jumlah maksimum kategori dalam fitur kategoris apa pun adalah 100. Jika sebuah properti memiliki lebih dari 100 kategori nilai, hanya 99 yang paling umum ditempatkan dalam kategori yang berbeda, dan sisanya ditempatkan dalam kategori khusus bernama`OTHER`.

Fitur numerik di Neptunus ML

Properti apa pun yang nilainya bilangan real dapat dikodekan sebagai fitur numerik di Neptunus ML. Fitur numerik dikodekan menggunakan angka floating-point.

Anda dapat menentukan metode normalisasi data yang akan digunakan saat menyandikan fitur numerik, seperti ini: `"norm": "`*`normalization technique`*`"` Teknik normalisasi berikut didukung:

- “tidak ada” — Jangan menormalkan nilai numerik selama pengkodean.
- “min-max” — Normalisasi setiap nilai dengan mengurangi nilai minimum darinya dan kemudian membaginya dengan selisih antara nilai maksimum dan minimum.
- “standar” — Menormalkan setiap nilai dengan membaginya dengan jumlah semua nilai.

Fitur bucket-numerik di Neptunus ML

Daripada mewakili properti numerik menggunakan angka mentah, Anda dapat memadatkan nilai numerik ke dalam kategori. Misalnya, Anda dapat membagi usia orang ke dalam kategori seperti anak-anak (0-20), dewasa muda (20-40), orang paruh baya (40-60) dan orang tua (dari 60 pada). Dengan menggunakan bucket numerik ini, Anda akan mengubah properti numerik menjadi semacam fitur kategoris.

Di Neptunus ML, Anda dapat menyebabkan properti numerik dikodekan sebagai fitur bucket-numerik, Anda harus memberikan dua hal:

- Rentang numerik dalam bentuk, `"range": [`*`a`*`, `*`b`*`]` , di mana a dan b merupakan bilangan bulat.
- Hitungan ember, dalam bentuk `"bucket_cnt": `*`c`* , di c mana jumlah ember, juga bilangan bulat.

Neptunus ML kemudian menghitung ukuran setiap bucket `( b - a ) / c` sebagai, dan mengkodekan setiap nilai numerik sebagai jumlah bucket apa pun yang masuk ke dalamnya. Nilai apa pun yang kurang dari a dianggap termasuk dalam ember pertama, dan nilai apa pun yang lebih besar dari b dianggap termasuk dalam ember terakhir.

Anda juga dapat, secara opsional, membuat nilai numerik jatuh ke lebih dari satu ember, dengan menentukan ukuran jendela geser, seperti ini: `"slide_window_size": s` , di mana angka. `s` Neptunus ML kemudian mengubah setiap `v` nilai numerik properti menjadi rentang `v - s/2` dari `v + s/2` hingga, dan memberikan `v` nilai ke setiap ember yang dicakup oleh rentang tersebut.

Terakhir, Anda juga dapat secara opsional menyediakan cara mengisi nilai yang hilang untuk fitur numerik dan fitur bucket-numerik. Anda melakukan ini menggunakan `"imputer": "imputation technique "`, di mana teknik imputasi adalah salah satu dari`"mean"`,`"median"`, atau. `"most-frequent"` Jika Anda tidak menentukan imputer, nilai yang hilang dapat menyebabkan pemrosesan terhenti.

Pengkodean fitur teks di Neptunus ML

Untuk teks bentuk bebas, Neptunus ML dapat menggunakan beberapa model berbeda untuk mengonversi urutan token dalam string nilai properti menjadi vektor nilai nyata ukuran tetap:

- `text_fasttext`— Menggunakan pengkodean FastText. Ini adalah pengkodean yang disarankan untuk fitur yang menggunakan satu dan hanya satu dari lima bahasa yang didukung FastText.
- `text_sbert`— Menggunakan model pengkodean Sentence BERT (SBERT). Ini adalah pengkodean yang disarankan untuk teks yang `text_fasttext` tidak mendukung.
- `text_word2vec`— Menggunakan algoritma Word2Vec awalnya diterbitkan oleh Google untuk menyandikan teks. Word2Vec hanya mendukung bahasa Inggris.
- `text_tfidf`— Menggunakan istilah frekuensi—inverse document frequency (TF-IDF) vectorizer untuk pengkodean teks. Pengkodean TF-IDF mendukung fitur statistik yang tidak dilakukan pengkodean lainnya.

FastText encoding nilai properti teks di Neptunus ML

Neptunus ML dapat menggunakan model FastText untuk mengubah nilai properti teks menjadi vektor nilai nyata ukuran tetap. Ini adalah metode pengkodean yang direkomendasikan untuk nilai properti teks di salah satu dari lima bahasa yang didukung FastText:

- `en`(Bahasa Inggris)

- `zh`(Tionghoa)
- `hi`(Hindi)
- `es`(Spanyol)
- `fr`(Perancis)

Perhatikan bahwa FastText tidak dapat menangani kalimat yang mengandung kata-kata dalam lebih dari satu bahasa.

`text_fasttext`Metode ini opsional dapat mengambil `max_length` bidang yang menentukan jumlah maksimum token dalam nilai properti teks yang akan dikodekan, setelah itu string terpotong. Ini dapat meningkatkan kinerja ketika nilai properti teks berisi string panjang, karena jika tidak `max_length` ditentukan, FastText mengkodekan semua token terlepas dari panjang string.

Contoh ini menentukan bahwa judul film Prancis dikodekan menggunakan FastText:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
        "feature": ["title", "title", "text_fasttext"],
        "language": "fr",
        "max_length": 1024
      }
    ]
  }
```

Sentence BERT (SBERT) encoding fitur teks di Neptunus ML

Neptunus ML dapat mengubah urutan token dalam nilai properti string menjadi vektor nilai nyata ukuran tetap menggunakan model Sentence BERT (SBERT). Neptunus mendukung dua metode SBERT`text_sbert128`:, yang merupakan default jika Anda hanya menentukan, dan. `text_sbert` `text_sbert512` Perbedaan antara keduanya adalah panjang maksimum string nilai properti teks yang dikodekan. `text_sbert128`Pengkodean memotong string teks setelah menyandikan 128 token, sementara `text_sbert512` memotong string teks setelah menyandikan 512 token. Akibatnya, `text_sbert512` membutuhkan lebih banyak waktu pemrosesan daripada`text_sbert128`. Kedua metode lebih lambat dari`text_fasttext`.

Pengkodean SBERT bersifat multibahasa, jadi tidak perlu menentukan bahasa untuk teks nilai properti yang Anda kodekan. SBERT mendukung banyak bahasa, dan dapat menyandikan kalimat yang berisi lebih dari satu bahasa. Jika Anda menyandikan nilai properti yang berisi teks dalam bahasa atau bahasa yang tidak didukung oleh FastText, SBERT adalah metode pengkodean yang disarankan.

Contoh berikut menentukan bahwa judul film dikodekan sebagai SBERT hingga maksimum 128 token:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      { "feature": ["title", "title", "text_sbert128"] }
    ]
  }
```

Pengkodean Word2Vec fitur teks di Neptunus ML

Neptunus ML dapat menyandikan nilai properti string sebagai fitur Word2Vec (algoritma Word2Vec awalnya diterbitkan oleh Google). `text_word2vec`Metode ini mengkodekan token dalam string sebagai vektor padat menggunakan salah satu model terlatih SPacy. Ini hanya mendukung bahasa Inggris menggunakan model en_core_web_lg).

Contoh berikut menentukan bahwa judul film dikodekan menggunakan Word2Vec:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
        "feature": ["title", "title", "text_word2vec"],
        "language": "en_core_web_lg"
      }
    ]
  }
```

Perhatikan bahwa bidang bahasa adalah opsional, karena `en_core_web_lg` model bahasa Inggris adalah satu-satunya yang didukung Neptunus.

Pengkodean fitur teks TF-IDF di Neptunus

Neptunus ML dapat menyandikan nilai properti teks sebagai fitur. `text_tfidf` Pengkodean ini mengubah urutan kata dalam teks menjadi vektor numerik menggunakan istilah frekuensi-inverse document frequency (TF-IDF) vectorizer, diikuti oleh operasi reduksi dimensi.

TF-IDF (frekuensi istilah — frekuensi dokumen terbalik) adalah nilai numerik yang dimaksudkan untuk mengukur seberapa penting sebuah kata dalam kumpulan dokumen. Ini dihitung dengan membagi berapa kali sebuah kata muncul dalam nilai properti tertentu dengan jumlah total nilai properti yang muncul di.

Misalnya, jika kata “ciuman” muncul dua kali dalam judul film tertentu (katakanlah, “ciuman ciuman bang bang”), dan “ciuman” muncul dalam judul 4 film secara keseluruhan, maka nilai TF-IDF dari “ciuman” dalam judul “ciuman ciuman bang” akan menjadi. `2 / 4`

Vektor yang awalnya dibuat memiliki dimensi d, di mana d adalah jumlah istilah unik di semua nilai properti dari jenis itu. Operasi pengurangan dimensi menggunakan proyeksi jarang acak untuk mengurangi angka itu hingga maksimum 100. Kosakata grafik kemudian dihasilkan dengan menggabungkan semua `text_tfidf` fitur di dalamnya.

Anda dapat mengontrol vectorizer TF-IDF dengan beberapa cara:

- **`max_features`**— Dengan menggunakan `max_features` parameter, Anda dapat membatasi jumlah istilah dalam `text_tfidf` fitur ke yang paling umum. Misalnya, jika Anda menyetel `max_features` ke 100, hanya 100 istilah teratas yang paling umum digunakan yang disertakan. Nilai default untuk `max_features` jika Anda tidak secara eksplisit menyetelnya adalah 5.000.
- **`min_df`**— Menggunakan `min_df` parameter, Anda dapat membatasi jumlah istilah dalam `text_tfidf` fitur untuk yang memiliki setidaknya frekuensi dokumen tertentu. Misalnya, jika Anda menyetel `min_df` ke 5, hanya istilah yang muncul di setidaknya 5 nilai properti yang berbeda yang digunakan. Nilai default untuk `min_df` jika Anda tidak secara eksplisit mengaturnya adalah 2.
- **`ngram_range`**— `ngram_range` Parameter menentukan kombinasi kata apa yang diperlakukan sebagai istilah. Misalnya, jika Anda mengatur `ngram_range` ke`[2, 4]`, 6 istilah berikut akan ditemukan di judul “kiss kiss bang bang”:
  - Istilah 2 kata: “ciuman ciuman”, “ciuman bang”, dan “bang bang”.
  - Istilah 3 kata: “kiss kiss bang” dan “kiss bang bang”.
  - Istilah 4 kata: “ciuman ciuman bang bang”.

  Pengaturan default untuk `ngram_range` adalah`[1, 1]`.

Fitur Datetime di Neptunus ML

Neptunus ML dapat mengubah bagian `datetime` dari nilai properti menjadi fitur kategoris dengan menyandikannya sebagai array satu-panas. Gunakan `datetime_parts` parameter untuk menentukan satu atau beberapa bagian berikut untuk menyandikan:`["year", "month", "weekday", "hour"]`. Jika Anda tidak mengaturdatetime_parts, secara default keempat bagian dikodekan.

Misalnya, jika rentang nilai datetime mencakup tahun 2010 hingga 2012, empat bagian dari entri datetime adalah sebagai berikut: `2011-04-22 01:16:34`

- tahun —`[0, 1, 0]`.

  Karena hanya ada 3 tahun dalam rentang (2010, 2011, dan 2012), array satu-panas memiliki tiga entri, satu untuk setiap tahun.
- bulan —`[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]`.

  Di sini, array satu-panas memiliki entri untuk setiap bulan dalam setahun.
- hari kerja —`[0, 0, 0, 0, 1, 0, 0]`.

  Standar ISO 8601 menyatakan bahwa Senin adalah hari pertama dalam seminggu, dan sejak 22 April 2011 adalah hari Jumat, array hari kerja satu panas yang sesuai panas di posisi kelima.
- jam —`[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`.

  Jam 1 pagi diatur dalam array satu-panas 24 anggota.

Hari dalam sebulan, menit, dan detik tidak dikodekan secara kategoris.

Jika `datetime` rentang total yang dimaksud hanya mencakup tanggal dalam satu tahun, tidak ada `year` array yang dikodekan.

Anda dapat menentukan strategi imputasi untuk mengisi `datetime` nilai yang hilang, menggunakan `imputer` parameter dan salah satu strategi yang tersedia untuk fitur numerik.

Pengkodean fitur otomatis di Neptunus ML

Alih-alih secara manual menentukan metode pengkodean fitur yang akan digunakan untuk properti dalam grafik Anda, Anda dapat mengatur `auto` sebagai metode pengkodean fitur. Neptunus ML

kemudian mencoba menyimpulkan pengkodean fitur terbaik untuk setiap properti berdasarkan tipe data yang mendasarinya.

Berikut adalah beberapa heuristik yang digunakan Neptunus dalam memilih pengkodean fitur yang sesuai:

- Jika properti hanya memiliki nilai numerik dan dapat dilemparkan ke tipe data numerik, maka Neptunus ML umumnya mengkodekannya sebagai nilai numerik. Namun, jika jumlah nilai unik untuk properti kurang dari 10% dari jumlah total nilai dan kardinalitas nilai unik tersebut kurang dari 100, maka Neptunus ML menggunakan pengkodean kategoris.
- Jika nilai properti dapat dilemparkan ke suatu `datetime` tipe, maka Neptunus ML mengkodekannya sebagai fitur. `datetime`
- Jika nilai properti dapat dipaksa untuk boolean (1/0 atau True/False), maka Neptunus ML menggunakan kategori encoding.
- Jika properti adalah string dengan lebih dari 10% nilainya unik, dan jumlah rata-rata token per nilai lebih besar dari atau sama dengan 3, Neptunus ML menyimpulkan jenis properti menjadi teks dan secara otomatis mendeteksi bahasa yang digunakan. Jika bahasa yang terdeteksi adalah salah satu yang didukung oleh FastText, yaitu Inggris, Mandarin, Hindi, Spanyol dan Prancis, maka Neptunus menggunakan untuk menyandikan teks. `text_fasttext` Sebaliknya, penggunaan Neptunus ML. `text_sbert`
- Jika properti adalah string yang tidak diklasifikasikan sebagai fitur teks maka Neptunus ML menganggapnya sebagai fitur kategoris dan menggunakan pengkodean kategori.
- Jika setiap node memiliki nilai uniknya sendiri untuk properti yang disimpulkan sebagai fitur kategori, Neptunus ML menjatuhkan properti dari grafik pelatihan karena mungkin ID yang tidak informatif untuk pembelajaran.
- Jika properti diketahui mengandung pemisah Neptunus yang valid seperti titik koma (";"), maka Neptunus ML hanya dapat memperlakukan properti sebagai atau. `MultiNumerical` `MultiCategorical`
  - Neptunus ML pertama kali mencoba untuk menyandikan nilai sebagai fitur numerik. jika ini berhasil, Neptunus ML menggunakan pengkodean numerik untuk membuat fitur vektor numerik.
  - Jika tidak, Neptunus ML mengkodekan nilai sebagai multi-kategoris.
- Jika Neptunus ML tidak dapat menyimpulkan tipe data dari nilai properti, MLdrops Neptunus properti dari grafik pelatihan.

Mengedit file konfigurasi data pelatihan

Proses ekspor Neptune mengekspor data Neptune ML dari klaster DB Neptune ke dalam bucket S3. Proses ini mengekspor simpul dan edge secara terpisah menjadi `nodes/` dan sebuah folder `edges/`. Ini juga membuat file konfigurasi data pelatihan JSON, dinamai secara `training-data-configuration.json` default. File ini berisi informasi tentang skema grafik, jenis fitur-fiturnya, transformasi fitur dan operasi normalisasi, dan fitur target untuk klasifikasi atau tugas regresi.

Mungkin ada kasus ketika Anda ingin memodifikasi file konfigurasi secara langsung. Salah satu kasus tersebut adalah ketika Anda ingin mengubah cara fitur diproses atau bagaimana grafik dibangun, tanpa perlu menjalankan kembali ekspor setiap kali Anda ingin memodifikasi spesifikasi untuk tugas pembelajaran mesin yang Anda selesaikan.

Untuk mengedit file konfigurasi data pelatihan

1. Unduh file ke mesin lokal Anda.

   Kecuali Anda menentukan satu atau lebih pekerjaan bernama dalam `additionalParams/neptune_ml` parameter yang diteruskan ke proses ekspor, file akan memiliki nama default, yaitu`training-data-configuration.json`. Anda dapat menggunakan perintah AWS CLI seperti ini untuk mengunduh file:

   ```
   aws s3 cp \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json \
     ./
   ```

2. Edit file menggunakan editor teks.
3. Unggah file yang dimodifikasi. Unggah file yang dimodifikasi kembali ke lokasi yang sama di Amazon S3 tempat Anda mengunduhnya, menggunakan perintah AWS CLI seperti ini:

   ```
   aws s3 cp \
     training-data-configuration.json \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json
   ```

Contoh file konfigurasi data pelatihan JSON

Berikut adalah contoh file konfigurasi data pelatihan yang menjelaskan grafik untuk tugas klasifikasi simpul:

```
{
  "version" : "v2.0",
  "query_engine" : "gremlin",
  "graph" : [
    {
      "edges" : [
        {
          "file_name" : "edges/(movie)-included_in-(genre).csv",
          "separator" : ",",
          "source" : ["~from", "movie"],
          "relation" : ["", "included_in"],
          "dest" : [ "~to", "genre" ]
        },
        {
          "file_name" : "edges/(user)-rated-(movie).csv",
          "separator" : ",",
          "source" : ["~from", "movie"],
          "relation" : ["rating", "prefixname"], # [prefixname#value]
          "dest" : ["~to", "genre"],
          "features" : [
            {
              "feature" : ["rating", "rating", "numerical"],
              "norm" : "min-max"
            }
          ]
        }
      ],
      "nodes" : [
        {
          "file_name" : "nodes/genre.csv",
          "separator" : ",",
          "node" : ["~id", "genre"],
          "features" : [
            {
              "feature": ["name", "genre", "category"],
              "separator": ";"
            }
          ]
```

```
        },
        {
          "file_name" : "nodes/movie.csv",
          "separator" : ",",
          "node" : ["~id", "movie"],
          "features" : [
            {
              "feature": ["title", "title", "word2vec"],
              "language": ["en_core_web_lg"]
            }
          ]
        },
        {
          "file_name" : "nodes/user.csv",
          "separator" : ",",
          "node" : ["~id", "user"],
          "features" : [
            {
              "feature": ["age", "age", "numerical"],
              "norm" : "min-max",
              "imputation": "median",
            },
            {
              "feature": ["occupation", "occupation", "category"],
            }
          ],
          "labels" : [
            {
              "label": ["gender", "classification"],
              "split_rate" : [0.8, 0.2, 0.0]
            }
          ]
        }
      ]
    },
    "warnings" : [ ]
  ]
}
```

Struktur file konfigurasi data pelatihan JSON

File konfigurasi pelatihan mengacu pada file CSV yang disimpan oleh proses ekspor di folder `nodes/` dan `edges/`.

Setiap berkas di bawah informasi penyimpanan `nodes/` tentang simpul yang memiliki label simpul grafik properti yang sama. Setiap kolom di file simpul menyimpan baik ID simpul ataupun properti simpul. Baris pertama dari file berisi header yang menentukan `~id` atau nama properti untuk setiap kolom.

Setiap berkas di bawah `edges/` menyimpan informasi tentang simpul yang memiliki label simpul grafik properti yang sama. Setiap kolom di file simpul menyimpan baik ID simpul sumber atau properti edge. Baris pertama dari file berisi header yang menentukan `~from`, `~to`, atau nama properti untuk setiap kolom.

File konfigurasi data pelatihan memiliki tiga elemen tingkat atas:

```
{
  "version" : "v2.0",
  "query_engine" : "gremlin",
  "graph" : [ ... ]
}
```

- `version`— (String) Versi file konfigurasi yang digunakan.
- `query_engine`— (String) Bahasa query yang digunakan untuk mengekspor data grafik. Saat ini, hanya “gremlin” yang valid.
- `graph`— (JSON array) mencantumkan satu atau lebih objek konfigurasi yang berisi parameter model untuk masing-masing node dan tepi yang akan digunakan.

  Objek konfigurasi dalam array grafik memiliki struktur yang dijelaskan di bagian berikutnya.

Isi dari objek konfigurasi yang tercantum dalam **`graph`** array

Objek konfigurasi dalam `graph` array dapat berisi tiga node tingkat atas:

```
    {
      "edges"    : [ ... ],
      "nodes"    : [ ... ],
      "warnings" : [ ... ],
    }
```

- `edges`— (array objek JSON) Setiap objek JSON menentukan satu set parameter untuk menentukan bagaimana tepi dalam grafik akan diperlakukan selama pemrosesan model dan pelatihan. Ini hanya digunakan dengan mesin Gremlin.

- `nodes`— (array objek JSON) Setiap objek JSON menentukan satu set parameter untuk menentukan bagaimana node dalam grafik akan diperlakukan selama pemrosesan model dan pelatihan. Ini hanya digunakan dengan mesin Gremlin.
- `warnings`— (array objek JSON) Setiap objek berisi peringatan yang dihasilkan selama proses ekspor data.

Isi dari objek konfigurasi tepi yang tercantum dalam **`edges`** array

Objek konfigurasi tepi yang tercantum dalam `edges` array dapat berisi bidang tingkat atas berikut:

```
    {
      "file_name" : "(path to a CSV file)",
      "separator" : "(separator character)",
      "source"    : ["(column label for starting node ID)", "(starting node type)"],
      "relation"  : ["(column label for the relationship name)", "(the prefix name
for the relationship name)"],
      "dest"      : ["(column label for ending node ID)", "(ending node type)"],
      "features"  : [(array of feature objects)],
      "labels"    : [(array of label objects)]
    }
```

- **`file_name`**— String yang menentukan jalur ke file CSV yang menyimpan informasi tentang tepi yang memiliki label grafik properti yang sama.

  Baris pertama dari file itu berisi baris header label kolom.

  Dua label kolom pertama adalah `~from` dan`~to`. Kolom pertama (`~from`kolom) menyimpan ID dari simpul awal tepi, dan yang kedua (`~to`kolom) menyimpan ID dari simpul akhir tepi.

  Label kolom yang tersisa di baris header menentukan, untuk setiap kolom yang tersisa, nama properti tepi yang nilainya telah diekspor ke kolom itu.
- **`separator`**— String yang berisi pembatas yang memisahkan kolom dalam file CSV itu.
- **`source`**- Array JSON yang berisi dua string yang menentukan simpul awal tepi. String pertama berisi nama header kolom tempat ID simpul awal disimpan. String kedua menentukan jenis node.
- **`relation`**— Sebuah array JSON yang berisi dua string yang menentukan tipe relasi edge. String pertama berisi nama header kolom tempat nama relasi (`relname`) disimpan. String kedua berisi awalan untuk nama relasi (`prefixname`).

Tipe relasi penuh terdiri dari dua string yang digabungkan, dengan karakter tanda hubung di antara keduanya, seperti ini:. *`prefixname`*`-`*`relname`*

Jika string pertama kosong, semua tepi memiliki tipe relasi yang sama, yaitu `prefixname` string.

- **`dest`**- Array JSON yang berisi dua string yang menentukan simpul akhir tepi. String pertama berisi nama header kolom tempat ID node disimpan. String kedua menentukan jenis node.
- **`features`**— Sebuah array JSON dari objek fitur nilai properti. Setiap objek fitur nilai properti berisi bidang-bidang berikut:
  - fitur - Sebuah array JSON dari tiga string. String pertama berisi nama header kolom yang berisi nilai properti. String kedua berisi nama fitur. String ketiga berisi jenis fitur.
  - norma - (Opsional) Menentukan metode normalisasi untuk diterapkan pada nilai-nilai properti.
- **`labels`**— Sebuah array JSON objek. Masing-masing objek mendefinisikan fitur target dari tepi, dan menentukan proporsi tepi yang harus diambil oleh tahap pelatihan dan validasi. Setiap objek berisi bidang-bidang berikut:
  - label - Sebuah array JSON dari dua string. String pertama berisi nama header kolom yang berisi nilai properti fitur target. String kedua menentukan salah satu jenis tugas target berikut:
    - `"classification"`— Tugas klasifikasi tepi. Nilai properti yang disediakan dalam kolom yang diidentifikasi oleh string pertama dalam `label` array diperlakukan sebagai nilai kategoris. Untuk tugas klasifikasi tepi, string pertama dalam `label` array tidak bisa kosong.
    - `"regression"`— Tugas regresi tepi. Nilai properti yang disediakan dalam kolom yang diidentifikasi oleh string pertama dalam `label` array diperlakukan sebagai nilai numerik. Untuk tugas regresi tepi, string pertama dalam `label` array tidak bisa kosong.
    - `"link_prediction"`— Tugas prediksi tautan. Tidak ada nilai properti yang diperlukan. Untuk tugas prediksi tautan, string pertama dalam `label` array diabaikan.
  - **`split_rate`**— Array JSON yang berisi tiga angka antara nol dan satu yang menambahkan hingga satu dan yang mewakili perkiraan proporsi node yang akan digunakan masing-masing tahap pelatihan, validasi, dan pengujian. Entah bidang ini atau `custom_split_filenames` dapat didefinisikan, tetapi tidak keduanya. Lihat split_rate.
  - **`custom_split_filenames`**— Objek JSON yang menentukan nama file untuk file yang menentukan populasi pelatihan, validasi, dan pengujian. Entah bidang ini atau `split_rate` dapat didefinisikan, tetapi tidak keduanya. Untuk informasi selengkapnya, lihat train-validation-testProporsi khusus.

Isi dari objek konfigurasi node yang tercantum dalam **`nodes`** array

Sebuah objek konfigurasi node yang tercantum dalam `nodes` array dapat berisi bidang-bidang berikut:

```
    {
      "file_name" : "(path to a CSV file)",
      "separator" : "(separator character)",
      "node"      : ["(column label for the node ID)", "(node type)"],
      "features"  : [(feature array)],
      "labels"    : [(label array)],
    }
```

- **`file_name`**— String yang menentukan jalur ke file CSV yang menyimpan informasi tentang node yang memiliki label grafik properti yang sama.

  Baris pertama dari file itu berisi baris header label kolom.

  Label kolom pertama adalah`~id`, dan kolom pertama (`~id`kolom) menyimpan ID node.

  Label kolom yang tersisa di baris header menentukan, untuk setiap kolom yang tersisa, nama properti node yang nilainya telah diekspor ke kolom itu.
- **`separator`**— String yang berisi pembatas yang memisahkan kolom dalam file CSV itu.
- **`node`**— Sebuah array JSON yang berisi dua string. String pertama berisi nama header kolom yang menyimpan node IDs. String kedua menentukan jenis node dalam grafik, yang sesuai dengan label property-graph dari node.
- **`features`**- Sebuah array JSON dari objek fitur node. Lihat Isi objek fitur yang tercantum dalam `features` array untuk simpul atau tepi.
- **`labels`**- Sebuah array JSON dari objek label node. Lihat Isi dari objek label node yang tercantum dalam `labels` array node.

Isi objek fitur yang tercantum dalam **`features`** array untuk simpul atau tepi

Objek fitur node yang terdaftar dalam `features` array node dapat berisi bidang tingkat atas berikut:

- **`feature`**— Sebuah array JSON dari tiga string. String pertama berisi nama header kolom yang berisi nilai properti untuk fitur tersebut. String kedua berisi nama fitur.

String ketiga berisi jenis fitur. Jenis fitur yang valid tercantum dalamNilai yang mungkin dari bidang tipe untuk fitur.

- **`norm`**- Bidang ini diperlukan untuk fitur numerik. Ini menentukan metode normalisasi untuk digunakan pada nilai-nilai numerik. Nilai yang valid adalah`"none"`,`"min-max"`, dan “standar”. Lihat Bidang norma untuk detail.
- **`language`**- Bidang bahasa menentukan bahasa yang digunakan dalam nilai properti teks. Penggunaannya tergantung pada metode pengkodean teks:
  - Untuk `text_fasttext`pengkodean, bidang ini diperlukan, dan harus menentukan salah satu bahasa berikut:
    - `en`(Bahasa Inggris)
    - `zh`(Tionghoa)
    - `hi`(Hindi)
    - `es`(Spanyol)
    - `fr`(Perancis)

    Namun, `text_fasttext` tidak dapat menangani lebih dari satu bahasa dalam satu waktu.
  - Untuk `text_sbert`pengkodean, bidang ini tidak digunakan, karena pengkodean SBERT bersifat multibahasa.
  - Untuk `text_word2vec`pengkodean, bidang ini opsional, karena `text_word2vec` hanya mendukung bahasa Inggris. Jika ada, itu harus menentukan nama model bahasa Inggris:

    ```
    "language" : "en_core_web_lg"
    ```
  - Untuk `tfidf`pengkodean, bidang ini tidak digunakan.
- **`max_length`**— Bidang ini opsional untuk `text_fasttext`fitur, di mana ia menentukan jumlah maksimum token dalam fitur teks input yang akan dikodekan. Teks masukan setelah `max_length` tercapai diabaikan. Misalnya, menyetel max_length ke 128 menunjukkan bahwa token apa pun setelah ke-128 dalam urutan teks diabaikan.
- **`separator`**— Bidang ini digunakan secara opsional dengan`category`, `numerical` dan `auto` fitur. Ini menentukan karakter yang dapat digunakan untuk membagi nilai properti menjadi beberapa nilai kategoris atau nilai numerik.

  Lihat Bidang pemisah.

- **`range`**- Bidang ini diperlukan untuk `bucket_numerical` fitur. Ini menentukan kisaran nilai numerik yang akan dibagi menjadi ember.

  Lihat Bidang rentang.

- **`bucket_cnt`**- Bidang ini diperlukan untuk `bucket_numerical` fitur. Ini menentukan jumlah ember yang rentang numerik yang ditentukan oleh `range` parameter harus dibagi menjadi.

  Lihat Fitur bucket-numerik di Neptunus ML.

- **`slide_window_size`**— Bidang ini digunakan secara opsional dengan `bucket_numerical` fitur untuk menetapkan nilai ke lebih dari satu bucket.

  Lihat Bidang slide_window_size.

- **`imputer`**— Bidang ini digunakan secara opsional dengan`numerical`,`bucket_numerical`, dan `datetime` fitur untuk memberikan teknik imputasi untuk mengisi nilai yang hilang. Teknik imputasi yang didukung adalah`"mean"`,`"median"`, dan. `"most_frequent"`

  Lihat Bidang imputer.

- **`max_features`**— Bidang ini digunakan secara opsional oleh `text_tfidf` fitur untuk menentukan jumlah maksimum istilah untuk dikodekan.

  Lihat Bidang max_features.

- **`min_df`**— Bidang ini digunakan secara opsional oleh `text_tfidf` fitur untuk menentukan frekuensi dokumen minimum istilah untuk dikodekan

  Lihat Bidang min_df.

- **`ngram_range`**— Bidang ini digunakan secara opsional oleh `text_tfidf` fitur untuk menentukan rentang jumlah kata atau token untuk dianggap sebagai istilah individu potensial untuk dikodekan

  Lihat Bidang ngram_range.

- **`datetime_parts`**— Bidang ini digunakan secara opsional oleh `datetime` fitur untuk menentukan bagian mana dari nilai datetime yang akan dikodekan secara kategoris.

  Lihat Bidang datetime_parts.

Isi dari objek label node yang tercantum dalam **`labels`** array node

Objek label yang tercantum dalam `labels` array node mendefinisikan fitur target node dan menentukan proporsi node yang akan digunakan oleh tahap pelatihan, validasi, dan pengujian. Setiap objek dapat berisi bidang-bidang berikut:

```
    {
      "label"      : ["(column label for the target feature property value)", "(task
type)"],
      "split_rate" : [(training proportion), (validation proportion), (test
proportion)],
      "custom_split_filenames" : {"train": "(training file name)", "valid":
"(validation file name)", "test": "(test file name)"},
      "separator"  : "(separator character for node-classification category values)",
    }
```

- **`label`**— Sebuah array JSON yang berisi dua string. String pertama berisi nama header kolom yang menyimpan nilai properti untuk fitur tersebut. String kedua menentukan jenis tugas target, yang dapat berupa:
  - `"classification"`— Tugas klasifikasi simpul. Nilai properti di kolom tertentu digunakan untuk membuat fitur kategoris.
  - `"regression"`— Tugas regresi simpul. Nilai properti di kolom yang ditentukan digunakan untuk membuat fitur numerik.
- **`split_rate`**— Array JSON yang berisi tiga angka antara nol dan satu yang menambahkan hingga satu dan mewakili perkiraan proporsi node yang akan digunakan masing-masing tahap pelatihan, validasi, dan pengujian. Lihat split_rate.
- **`custom_split_filenames`**— Objek JSON yang menentukan nama file untuk file yang menentukan populasi pelatihan, validasi, dan pengujian. Entah bidang ini atau `split_rate` dapat didefinisikan, tetapi tidak keduanya. Untuk informasi selengkapnya, lihat train-validation-testProporsi khusus.
- **`separator`**— String yang berisi pembatas yang memisahkan nilai fitur kategoris untuk tugas klasifikasi.

> **Note**
>
> Jika tidak ada objek label yang disediakan untuk kedua tepi dan node, tugas secara otomatis diasumsikan sebagai prediksi tautan, dan tepi secara acak dibagi menjadi 90% untuk pelatihan dan 10% untuk validasi.

train-validation-testProporsi khusus

Secara default, `split_rate` parameter digunakan oleh Neptunus ML untuk membagi grafik secara acak menjadi populasi pelatihan, validasi, dan uji menggunakan proporsi yang ditentukan dalam parameter ini. Untuk memiliki kontrol yang lebih tepat atas entitas mana yang digunakan dalam populasi yang berbeda ini, file dapat dibuat yang secara eksplisit mendefinisikannya, dan kemudian file konfigurasi data pelatihan dapat diedit untuk memetakan file pengindeksan ini ke populasi. Pemetaan ini ditentukan oleh objek JSON untuk `custom_split_filesnames`kunci dalam file konfigurasi pelatihan. Jika opsi ini digunakan, nama file harus disediakan untuk kunci dan, `train` dan opsional untuk `validation` kunci. `test`

Pemformatan file-file ini harus sesuai dengan format data Gremlin. Secara khusus, untuk tugas tingkat simpul, setiap file harus berisi kolom dengan `~id` header yang mencantumkan node IDs, dan untuk tugas tingkat tepi, file harus menentukan `~from` dan `~to` menunjukkan node sumber dan tujuan dari tepi, masing-masing. File-file ini harus ditempatkan di lokasi Amazon S3 yang sama dengan data yang diekspor yang digunakan untuk pemrosesan data (lihat:). `outputS3Path`

Untuk tugas klasifikasi atau regresi properti, file-file ini secara opsional dapat menentukan label untuk tugas pembelajaran mesin. Dalam hal ini file harus memiliki kolom properti dengan nama header yang sama seperti yang didefinisikan dalam file konfigurasi data pelatihan. Jika label properti didefinisikan dalam file node dan edge yang diekspor dan file split khusus, prioritas diberikan ke file split khusus.

Melatih model menggunakan Neptune ML

Setelah Anda memproses data yang Anda ekspor dari Neptune untuk pelatihan model, Anda dapat memulai tugas pelatihan model menggunakan perintah `curl` (atau `awscurl`) seperti berikut ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
      }'
```

Rincian tentang cara menggunakan perintah ini dijelaskan dalam Perintah modeltraining, bersama dengan informasi tentang cara untuk mendapatkan status tugas yang sedang berjalan, cara menghentikan tugas yang sedang berjalan, dan mendaftar semua tugas yang sedang berjalan.

Anda juga dapat menyediakan informasi penggunaan dari pekerjaan pelatihan model Neptunus ML yang telah selesai untuk mempercepat pencarian hyperparameter dalam pekerjaan pelatihan baru. `previousModelTrainingJobId` Ini berguna selama pelatihan ulang model pada data grafik baru, serta pelatihan tambahan pada data grafik yang sama. Gunakan perintah seperti ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "previousModelTrainingJobId" : "(the model-training job-id of a completed job)"
      }'
```

Anda dapat melatih implementasi model Anda sendiri pada infrastruktur pelatihan Neptunus Neptunus dengan memasok objek, `customModelTrainingParameters` seperti ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
```

```
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Lihat Perintah modeltraining untuk informasi selengkapnya, seperti tentang cara mendapatkan status pekerjaan yang sedang berjalan, cara menghentikan pekerjaan yang sedang berjalan, dan cara membuat daftar semua pekerjaan yang sedang berjalan. Lihat Model khusus di Neptunus ML untuk informasi tentang cara menerapkan dan menggunakan model kustom.

Topik

- Model dan pelatihan model di Amazon Neptunus
- Menyesuaikan konfigurasi hyperparameter model di Neptune ML
- Praktik terbaik pelatihan model

Model dan pelatihan model di Amazon Neptunus

Neptunus menggunakan Graph Neural Networks (GNN) untuk membuat model untuk berbagai tugas pembelajaran mesin. Jaringan saraf grafik telah terbukti memperoleh state-of-the-art hasil untuk tugas pembelajaran mesin grafik dan sangat baik dalam mengekstraksi pola informatif dari data terstruktur grafik.

Grafik jaringan saraf (GNNs) di Neptunus ML

Graph Neural Networks (GNNs) milik keluarga jaringan saraf yang menghitung representasi simpul dengan mempertimbangkan struktur dan fitur node terdekat. GNNs melengkapi pembelajaran mesin tradisional lainnya dan metode jaringan saraf yang tidak cocok untuk data grafik.

GNNs digunakan untuk menyelesaikan tugas-tugas pembelajaran mesin seperti klasifikasi node dan regresi (memprediksi properti node) dan klasifikasi tepi dan regresi (memprediksi properti tepi) atau prediksi tautan (memprediksi apakah dua node dalam grafik harus terhubung atau tidak).

Secara umum, menggunakan GNN untuk tugas pembelajaran mesin melibatkan dua tahap:

- Tahap pengkodean, di mana GNN menghitung vektor dimensi-d untuk setiap node dalam grafik. Vektor-vektor ini juga disebut representasi atau embeddings.
- Tahap decoding, yang membuat prediksi berdasarkan representasi yang dikodekan.

Untuk klasifikasi dan regresi simpul, representasi simpul digunakan secara langsung untuk tugas klasifikasi dan regresi. Untuk klasifikasi tepi dan regresi, representasi simpul dari node insiden di tepi digunakan sebagai masukan untuk klasifikasi atau regresi. Untuk prediksi tautan, skor kemungkinan tepi dihitung dengan menggunakan sepasang representasi simpul dan representasi tipe tepi.

Deep Graph Library (DGL) memfasilitasi definisi dan pelatihan yang efisien GNNs untuk tugas-tugas ini.

Model GNN yang berbeda disatukan di bawah perumusan pengiriman pesan. Dalam tampilan ini, representasi untuk node dalam grafik dihitung menggunakan representasi tetangga node (pesan), bersama dengan representasi awal node. Dalam Neptuneml, representasi awal dari sebuah node berasal dari fitur yang diekstraksi dari properti node, atau dapat dipelajari dan tergantung pada identitas node.

Neptunus ML juga menyediakan opsi untuk menggabungkan fitur simpul dan representasi simpul yang dapat dipelajari untuk berfungsi sebagai representasi simpul asli.

Untuk berbagai tugas di Neptunus ML yang melibatkan grafik dengan properti simpul, kami menggunakan Relational Graph Convolutional Network (R-GCN)) untuk melakukan tahap pengkodean. R-GCN adalah arsitektur GNN yang sangat cocok untuk grafik yang memiliki beberapa tipe simpul dan tepi (ini dikenal sebagai grafik heterogen).

Jaringan R-GCN terdiri dari sejumlah lapisan tetap, ditumpuk satu demi satu. Setiap lapisan R-GCN menggunakan parameter model yang dapat dipelajari untuk mengumpulkan informasi dari lingkungan 1-hop langsung dari sebuah node. Karena lapisan berikutnya menggunakan representasi keluaran lapisan sebelumnya sebagai input, radius lingkungan grafik yang mempengaruhi penyematan akhir node bergantung pada jumlah lapisan (`num-layer`), dari jaringan R-GCN.

Misalnya, ini berarti bahwa jaringan 2-layer menggunakan informasi dari node yang berjarak 2 hop.

Untuk mempelajari selengkapnya GNNs, lihat Survei Komprehensif tentang Jaringan Saraf Grafik. Untuk informasi lebih lanjut tentang Deep Graph Library (DGL), kunjungi halaman web DGL. Untuk tutorial langsung tentang penggunaan DGL dengan GNNs, lihat Mempelajari jaringan saraf grafik dengan Deep Graph Library.

Grafik Pelatihan Jaringan Saraf

Dalam pembelajaran mesin, proses mendapatkan model untuk belajar bagaimana membuat prediksi yang baik untuk suatu tugas disebut pelatihan model. Ini biasanya dilakukan dengan menentukan tujuan tertentu untuk dioptimalkan, serta algoritma yang digunakan untuk melakukan pengoptimalan ini.

Proses ini digunakan dalam pelatihan GNN untuk mempelajari representasi yang baik untuk tugas hilir juga. Kami membuat fungsi obyektif untuk tugas yang diminimalkan selama pelatihan model. Misalnya, untuk klasifikasi simpul, kami gunakan CrossEntropyLosssebagai tujuan, yang menghukum salah klasifikasi, dan untuk regresi simpul kami meminimalkan. MeanSquareError

Tujuannya biasanya fungsi kerugian yang mengambil prediksi model untuk titik data tertentu dan membandingkannya dengan nilai kebenaran dasar untuk titik data tersebut. Ini mengembalikan nilai kerugian, yang menunjukkan seberapa jauh prediksi model. Tujuan dari proses pelatihan adalah untuk meminimalkan kerugian dan memastikan bahwa prediksi model mendekati kebenaran dasar.

Algoritma optimasi yang digunakan dalam pembelajaran mendalam untuk proses pelatihan biasanya merupakan varian dari penurunan gradien. Di Neptunus ML, kami menggunakan Adam, yang merupakan algoritme untuk optimalisasi fungsi objektif stokastik berbasis gradien orde pertama berdasarkan perkiraan adaptif momen orde rendah.

Sementara proses pelatihan model mencoba untuk memastikan bahwa parameter model yang dipelajari mendekati minimum fungsi objektif, kinerja keseluruhan model juga tergantung pada hiperparameter model, yang merupakan pengaturan model yang tidak dipelajari oleh algoritma pelatihan. Misalnya, dimensi representasi simpul yang dipelajari,`num-hidden`, adalah hiperparameter yang memengaruhi kinerja model. Oleh karena itu, adalah umum dalam pembelajaran mesin untuk melakukan optimasi hyperparameter (HPO) untuk memilih hyperparameter yang sesuai.

Neptunus MLmenggunakan SageMaker pekerjaan tuning hyperparameter AI untuk meluncurkan beberapa contoh pelatihan model dengan konfigurasi hyperparameter yang berbeda untuk mencoba menemukan model terbaik untuk berbagai pengaturan hyperparameters. Lihat Menyesuaikan konfigurasi hyperparameter model di Neptune ML.

Model penyematan grafik pengetahuan di Neptunus ML

Grafik pengetahuan (KGs) adalah grafik yang mengkodekan informasi tentang entitas yang berbeda (node) dan hubungannya (tepi). Di Neptunus ML, model penyematan grafik pengetahuan diterapkan secara default untuk melakukan prediksi tautan ketika grafik tidak berisi properti simpul, hanya hubungan dengan node lain. Meskipun, model R-GCN dengan penyematan yang dapat dipelajari juga dapat digunakan untuk grafik ini dengan menentukan jenis model sebagai`"rgcn"`, model penyematan grafik pengetahuan lebih sederhana dan dirancang agar efektif untuk mempelajari representasi untuk grafik pengetahuan skala besar.

Model penyematan grafik pengetahuan digunakan dalam tugas prediksi tautan untuk memprediksi node atau hubungan yang menyelesaikan rangkap tiga `(h, r, t)` di mana **`h`** simpul sumber, **`r`** adalah tipe relasi dan **`t`** merupakan simpul tujuan.

Model penyematan grafik pengetahuan yang diterapkan di Neptunus ML `distmult` adalah,, dan. `transE` `rotatE` Untuk mempelajari lebih lanjut tentang model penyematan grafik pengetahuan, lihat DGL-KE.

Melatih model kustom di Neptunus ML

Neptunus ML memungkinkan Anda menentukan dan mengimplementasikan model kustom Anda sendiri, untuk skenario tertentu. Lihat Model khusus di Neptunus ML untuk informasi tentang cara menerapkan model kustom dan cara menggunakan infrastruktur Neptunus ML untuk melatihnya.

Menyesuaikan konfigurasi hyperparameter model di Neptune ML

Ketika Anda memulai tugas pelatihan model Neptune ML, Neptune ML otomatis menggunakan informasi yang disimpulkan dari tugas pemrosesan data sebelumnya. Ini menggunakan informasi untuk menghasilkan rentang konfigurasi hyperparameter yang digunakan untuk membuat pekerjaan tuning hyperparameter SageMaker AI untuk melatih beberapa model untuk tugas Anda. Dengan begitu, Anda tidak harus menentukan daftar panjang nilai hyperparameter untuk model yang akan dilatih. Sebaliknya, rentang dan default hyperparameter model dipilih berdasarkan jenis tugas, jenis grafik, dan pengaturan tugas penyetelan.

Namun, Anda juga dapat mengganti konfigurasi hyperparameter default dan menyediakan hyperparameters kustom dengan memodifikasi file konfigurasi JSON yang dihasilkan tugas pemrosesan data.

Menggunakan Neptune ML API Pelatihanmodel, Anda dapat mengontrol beberapa pengaturan tugas penyetelan hyperparameter tingkat tinggi seperti`maxHPONumberOfTrainingJobs`, `maxHPOParallelTrainingJobs`, dan `trainingInstanceType`. Untuk kontrol yang lebih halus atas hyperparameter model, Anda dapat menyesuaikan file `model-HPO-configuration.json` yang dihasilkan tugas pengolahan data. File disimpan di lokasi Amazon S3 yang Anda tentukan untuk output tugas pemrosesan.

Anda dapat mengunduh file, mengeditnya untuk mengganti konfigurasi hyperparameter default, dan mengunggahnya kembali ke lokasi Amazon S3 yang sama. Jangan mengubah nama file, dan hati-hati untuk mengikuti petunjuk ini saat Anda mengedit.

Mengunduh file dari Amazon S3.

```
aws s3 cp \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json \
  ./
```

Setelah selesai mengedit, unggah kembali file ke tempatnya:

```
aws s3 cp \
  model-HPO-configuration.json \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json
```

Struktur berkas **model-HPO-configuration.json**

File `model-HPO-configuration.json` menentukan model yang akan dilatih, machine learning `task_type` dan hyperparameters yang harus bervariasi atau diperbaiki untuk berbagai rangkaian pelatihan model.

Hyperparameters dikategorikan sebagai milik berbagai tier yang menandakan prioritas diberikan kepada hyperparameters ketika tugas penyetelan hyperparameter dipanggil:

- Hyperparameters Tier-1 memiliki prioritas tertinggi. Jika Anda mengatur `maxHPONumberOfTrainingJobs` ke nilai kurang dari 10, hanya hyperparameters Tier-1 yang disetel, dan sisanya mengambil nilai default mereka.
- Hyperparameters Tier-2 memiliki prioritas yang lebih rendah, jadi jika Anda memiliki lebih dari 10 tapi kurang dari 50 total tugas pelatihan untuk tugas penyetelan, maka hyperparameters Tier-1 dan Tier-2 disetel.
- Hyperparameters Tier 3 disetel bersama dengan Tier-1 dan Tier-2 hanya jika Anda memiliki lebih dari 50 total tugas pelatihan.
- Akhirnya, hiperparameter tetap tidak disetel sama sekali, dan selalu mengambil nilai defaultnya.

Contoh file **model-HPO-configuration.json**

Berikut ini adalah sampel file `model-HPO-configuration.json`:

```
{
  "models": [
    {
      "model": "rgcn",
      "task_type": "node_class",
      "eval_metric": {
        "metric": "acc"
      },
      "eval_frequency": {
          "type":  "evaluate_every_epoch",
          "value":  1
      },
      "1-tier-param": [
        {
            "param": "num-hidden",
            "range": [16, 128],
            "type": "int",
```

```
            "inc_strategy": "power2"
        },
        {
          "param": "num-epochs",
          "range": [3,30],
          "inc_strategy": "linear",
          "inc_val": 1,
          "type": "int",
          "node_strategy": "perM"
        },
        {
          "param": "lr",
          "range": [0.001,0.01],
          "type": "float",
          "inc_strategy": "log"
        }
      ],
      "2-tier-param": [
        {
          "param": "dropout",
          "range": [0.0,0.5],
          "inc_strategy": "linear",
          "type": "float",
          "default": 0.3
        },
        {
          "param": "layer-norm",
          "type": "bool",
          "default": true
        }
      ],
      "3-tier-param": [
        {
          "param": "batch-size",
          "range": [128, 4096],
          "inc_strategy": "power2",
          "type": "int",
          "default": 1024
        },
        {
          "param": "fanout",
          "type": "int",
          "options": [[10, 30],[15, 30], [15, 30]],
          "default": [10, 15, 15]
```

```
        },
        {
          "param": "num-layer",
          "range": [1, 3],
          "inc_strategy": "linear",
          "inc_val": 1,
          "type": "int",
          "default": 2
        },
        {
          "param": "num-bases",
          "range": [0, 8],
          "inc_strategy": "linear",
          "inc_val": 2,
          "type": "int",
          "default": 0
        }
      ],
      "fixed-param": [
        {
          "param": "concat-node-embed",
          "type": "bool",
          "default": true
        },
        {
          "param": "use-self-loop",
          "type": "bool",
          "default": true
        },
        {
          "param": "low-mem",
          "type": "bool",
          "default": true
        },
        {
          "param": "l2norm",
          "type": "float",
          "default": 0
        }
      ]
    }
  ]
}
```

Elemen file **`model-HPO-configuration.json`**

File berisi objek JSON dengan array tingkat atas tunggal bernama `models` yang berisi objek model-konfigurasi tunggal. Saat menyesuaikan file, pastikan `models` array hanya memiliki satu objek model-konfigurasi di dalamnya. Jika file Anda berisi lebih dari satu objek konfigurasi model, pekerjaan penyetelan akan gagal dengan peringatan.

Objek model-konfigurasi berisi elemen tingkat atas berikut:

- **`model`**— (String) Jenis model yang akan dilatih (jangan modifikasi). Nilai yang valid adalah:
  - `"rgcn"`— Ini adalah default untuk klasifikasi node dan tugas regresi, dan untuk tugas prediksi tautan heterogen.
  - `"transe"`— Ini adalah default untuk tugas prediksi tautan KGE.
  - `"distmult"`— Ini adalah jenis model alternatif untuk tugas prediksi tautan KGE.
  - `"rotate"`— Ini adalah jenis model alternatif untuk tugas prediksi tautan KGE.

  Sebagai aturan, jangan langsung memodifikasi nilai `model`, karena jenis model yang berbeda sering memiliki hyperparameter berlaku secara substansial berbeda, yang dapat mengakibatkan kesalahan parsing setelah pekerjaan pelatihan telah dimulai.

  Untuk mengubah jenis model, gunakan parameter `modelName` dalam modelTraining API ketimbang mengubahnya di file `model-HPO-configuration.json`.

  Cara untuk mengubah jenis model dan membuat perubahan hyperparameter fine-grain adalah dengan menyalin template konfigurasi model default untuk model yang ingin Anda gunakan dan menempelkannya ke file `model-HPO-configuration.json`. Ada folder bernama `hpo-configuration-templates` di lokasi Amazon S3 yang sama sebagai file `model-HPO-configuration.json` jika jenis tugas yang disimpulkan mendukung beberapa model. Folder ini berisi semua konfigurasi hyperparameter default untuk model lain yang berlaku untuk tugas tersebut.

  Misalnya, jika Anda ingin mengubah konfigurasi model dan hyperparameter untuk tugas prediksi link KGE dari default model `transe` ke model `distmult`, cukup tempelkan isi file `hpo-configuration-templates/distmult.json` ke file `model-HPO-configuration.json` dan kemudian edit hyperparameters seperti yang diperlukan.

> **Note**
>
> Jika Anda mengatur parameter `modelName` dalam API `modelTraining` dan juga mengubah spesifikasi hyperparameter `model` dalam file `model-HPO-configuration.json`, dan ini berbeda, nilai `model` dalam file `model-HPO-configuration.json` diutamakan, dan nilai `modelName` diabaikan.

- **`task_type`**— (String) Jenis tugas pembelajaran mesin yang disimpulkan oleh atau diteruskan langsung ke pekerjaan pemrosesan data (jangan modifikasi). Nilai yang valid adalah:
  - `"node_class"`
  - `"node_regression"`
  - `"link_prediction"`

  Tugas pemrosesan data menyimpulkan jenis tugas dengan memeriksa set data yang diekspor dan file konfigurasi tugas pelatihan yang dihasilkan untuk properti set data.

  Nilai ini seharusnya tidak diubah. Jika Anda ingin melatih tugas yang berbeda, Anda perlu menjalankan pekerjaan pemrosesan data baru. Jika nilai `task_type` adalah bukan yang Anda harapkan, Anda harus memeriksa input untuk tugas pemrosesan data Anda untuk memastikan bahwa mereka benar. Ini termasuk parameter untuk API `modelTraining`, sebagaimana dalam file konfigurasi tugas pelatihan yang dihasilkan oleh proses ekspor data.
- **`eval_metric`**— (String) Metrik evaluasi harus digunakan untuk mengevaluasi kinerja model dan untuk memilih model berkinerja terbaik di seluruh proses HPO. Nilai yang valid adalah:
  - `"acc"`- Akurasi klasifikasi standar. Ini adalah default untuk tugas klasifikasi label tunggal, kecuali label yang tidak seimbang ditemukan selama pemrosesan data, dalam hal ini defaultnya adalah. `"F1"`
  - `"acc_topk"`— Berapa kali label yang benar adalah salah satu **k**prediksi teratas. Anda juga dapat mengatur nilai **k**dengan meneruskan `topk` sebagai kunci tambahan.
  - `"F1"`— Skor F1.
  - `"mse"`— Metrik kesalahan kuadrat rata-rata, untuk tugas regresi.
  - `"mrr"`— Rata-rata metrik peringkat timbal balik.
  - `"precision"`— Presisi model, dihitung sebagai rasio positif sejati terhadap positif yang diprediksi:. `= true-positives / (true-positives + false-positives)`

- `"recall"`— Model recall, dihitung sebagai rasio positif sejati terhadap positif aktual:. `= true-positives / (true-positives + false-negatives)`
- `"roc_auc"`— Area di bawah kurva ROC. Ini adalah default untuk klasifikasi multi-label.

Misalnya, untuk mengubah metrik menjadiF1, ubah `eval_metric` nilainya sebagai berikut:

```
"  eval_metric": {
    "metric": "F1",
  },
```

Atau, untuk mengubah metrik menjadi skor `topk` akurasi, Anda akan mengubah `eval_metric` sebagai berikut:

```
  "eval_metric": {
    "metric": "acc_topk",
    "topk": 2
  },
```

- **`eval_frequency`**— (Objek) Menentukan seberapa sering selama pelatihan kinerja model pada set validasi harus diperiksa. Berdasarkan kinerja validasi, penghentian dini kemudian dapat dimulai dan model terbaik dapat disimpan.

  `eval_frequency`Objek tersebut mengandung dua elemen, yaitu `"type"` dan`"value"`. Contoh:

```
  "eval_frequency": {
    "type":  "evaluate_every_pct",
    "value":  0.1
  },
```

  `type`Nilai yang valid adalah:

  - **`evaluate_every_pct`**— Menentukan persentase pelatihan yang harus diselesaikan untuk setiap evaluasi.

    Untuk`evaluate_every_pct`, `"value"` bidang berisi angka floating-point antara nol dan satu yang menyatakan persentase itu.

  - **`evaluate_every_batch`**— Menentukan jumlah batch pelatihan yang harus diselesaikan untuk setiap evaluasi.

Untuk`evaluate_every_batch`, `"value"` bidang berisi integer yang menyatakan jumlah batch tersebut.

- **`evaluate_every_epoch`**— Menentukan jumlah epoch per evaluasi, di mana epoch baru dimulai pada tengah malam.

  Untuk`evaluate_every_epoch`, `"value"` bidang berisi bilangan bulat yang menyatakan jumlah epoch itu.

Pengaturan default untuk `eval_frequency` adalah:

```
"eval_frequency": {
  "type":  "evaluate_every_epoch",
  "value":  1
},
```

- **`1-tier-param`** – (Wajib) Sebuah array dari hyperparameters Tier-1 .

  Jika Anda tidak ingin menyetel hyperparameters apa pun, Anda dapat mengatur ini ke array kosong. Ini tidak mempengaruhi jumlah total pekerjaan pelatihan yang diluncurkan oleh pekerjaan tuning hyperparameter SageMaker AI. Ini hanya berarti bahwa semua tugas pelatihan, jika ada lebih dari 1 tapi kurang dari 10, maka akan berjalan dengan set hyperparameters yang sama.

  Di sisi lain, jika Anda ingin memperlakukan semua hyperparameters yang bisa disetel dengan signifikansi yang sama maka Anda dapat menempatkan semua hyperparameters dalam array ini.

- **`2-tier-param`** – (Wajib) Sebuah array dari hyperparameters Tier-2 .

  Parameter ini hanya disetel jika `maxHPONumberOfTrainingJobs` memiliki nilai lebih besar dari 10. Jika tidak, mereka tetap ke nilai default.

  Jika Anda memiliki anggaran pelatihan paling banyak 10 tugas pelatihan atau tidak menginginkan hyperparameters Tier-2 untuk alasan apapun, tetapi Anda ingin menyetel semua hyperparameters yang dapat disetel, maka Anda dapat mengatur ini ke array kosong.

- **`3-tier-param`** – (Wajib) Sebuah array dari hyperparameters Tier-3 .

  Parameter ini hanya disetel jika `maxHPONumberOfTrainingJobs` memiliki nilai lebih besar dari 50. Jika tidak, mereka tetap ke nilai default.

  Jika Anda tidak menginginkan hyperparameters Tier-3, Anda dapat mengatur ini ke array kosong.

- **`fixed-param`**— (Wajib) Array hyperparameters tetap yang hanya mengambil nilai defaultnya dan tidak mengambil bervariasi dalam tugas pelatihan yang berbeda.

  Jika Anda ingin memvariasi semua hyperparameter, Anda dapat mengatur ini ke array kosong dan juga mengatur nilai untuk `maxHPONumberOfTrainingJobs` cukup besar untuk memvariasi semua tier atau membuat semua hyperparameters Tier-1.

Objek JSON yang mewakili setiap hyperparameter di `1-tier-param`, `2-tier-param`, `3-tier-param`, dan `fixed-param` berisi elemen berikut:

- **`param`**— (String) Nama hyperparameter (jangan berubah).

  Lihat daftar nama hyperparameter yang valid di Neptune ML.
- **`type`**— (String) Tipe hyperparameter (jangan berubah).

  Jenis yang valid adalah: `bool`, `int`, dan `float`.
- **`default`**— (String) Nilai default untuk hyperparameter.

  Anda dapat mengatur nilai default baru.

Hyperparameters yang dapat disetel juga dapat berisi elemen berikut:

- **`range`**— (Array) Rentang untuk hyperparameter yang dapat disetel terus menerus.

  Ini seharusnya array dengan dua nilai, yaitu minimum dan maksimum rentang (`[min, max]`).
- **`options`**— (Array) Opsi untuk hyperparameter kategoris yang dapat disetel.

  Array ini harus berisi semua opsi untuk dipertimbangkan:

  ```
  "options" : [value1, value2, ... valuen]
  ```
- **`inc_strategy`**— (String) Jenis perubahan inkremental untuk rentang hyperparameter yang dapat disetel terus menerus (jangan berubah).

  Nilai yang valid adalah `log`, `linear`, dan `power2`. Ini hanya berlaku bila kunci rentang diatur.

  Memodifikasi kunci ini dapat mengakibatkan tidak digunakannya secara penuh hyperparameter Anda untuk penyetelan.

- **`inc_val`**— (Float) Jumlah dimana kenaikan berturut-turut berbeda untuk tunablehyperparameters kontinu (jangan berubah).

  Ini hanya berlaku bila kunci rentang diatur.

  Memodifikasi kunci ini dapat mengakibatkan tidak digunakannya secara penuh hyperparameter Anda untuk penyetelan.
- **`node_strategy`**— (String) Menunjukkan bahwa rentang efektif untuk hyperparameter ini harus berubah berdasarkan jumlah node dalam grafik (jangan berubah).

  Nilai yang valid adalah `"perM"` (per juta), `"per10M"` (per 10 juta), dan `"per100M"` (per 100 juta).

  Daripada mengubah nilai ini, ubah `range` sebagai gantinya.
- **`edge_strategy`**— (String) Menunjukkan bahwa rentang efektif untuk hyperparameter ini harus berubah berdasarkan jumlah tepi dalam grafik (jangan berubah).

  Nilai yang valid adalah `"perM"` (per juta), `"per10M"` (per 10 juta), dan `"per100M"` (per 100 juta).

  Daripada mengubah nilai ini, ubah `range` sebagai gantinya.

Daftar semua hyperparameters di Neptune ML

Daftar berikut berisi semua hyperparameters yang dapat diatur di mana saja di Neptune ML, untuk setiap jenis model dan tugas. Karena mereka semua tidak berlaku untuk setiap jenis model, adalah penting bahwa Anda hanya mengatur hyperparameters di file `model-HPO-configuration.json` yang muncul di templat untuk model yang Anda gunakan.

- **`batch-size`** – Ukuran batch simpul target yang digunakan dalam satu forward pass. Jenis:`int`.

  Mengatur ke nilai yang jauh lebih besar dapat menyebabkan masalah memori untuk pelatihan pada instans GPU.
- **`concat-node-embed`**— Menunjukkan apakah akan mendapatkan representasi awal dari sebuah node dengan menggabungkan fitur yang diproses dengan penyematan simpul awal yang dapat dipelajari untuk meningkatkan ekspresivitas model. Jenis:`bool`.
- **`dropout`** – Probabilitas dropout yang diterapkan pada lapisan dropout. Jenis:`float`.
- **`edge-num-hidden`**- Ukuran lapisan tersembunyi atau jumlah unit untuk modul fitur tepi. Hanya digunakan saat `use-edge-features` disetel ke`True`. Jenis: mengapung.

- **enable-early-stop**— Beralih apakah akan menggunakan fitur penghentian awal atau tidak. Jenis:`bool`. Default: `true`.

  Gunakan parameter Boolean ini untuk mematikan fitur berhenti awal.

- **fanout** – Jumlah neighbor yang akan dijadikan sampel untuk simpul target selama sampling neighbor. Jenis:`int`.

  Nilai ini digabungkan dengan erat dengan `num-layers` dan harus selalu berada di tier hyperparameter yang sama. Hal ini karena Anda dapat menentukan fanout untuk setiap layer GNN potensial.

  Karena hyperparameter ini dapat menyebabkan kinerja model bervariasi luas, seharusnya hyperparameter ini diperbaiki atau ditetapkan sebagai hyperparameter Tier-2 atau Tier-3. Mengaturnya ke nilai yang besar dapat menyebabkan masalah memori untuk pelatihan pada instans GPU.

- **gamma** – Nilai margin dalam fungsi skor. Jenis:`float`.

  Hal ini berlaku untuk model prediksi link `KGE` saja.

- **l2norm**- Nilai peluruhan berat yang digunakan dalam pengoptimal yang memberlakukan penalti normalisasi L2 pada bobot. Jenis:`bool`.
- **layer-norm** – Mengindikasikan apakah akan menggunakan normalisasi lapisan untuk model `rgcn`. Jenis:`bool`.
- **low-mem** – Mengindikasikan apakah akan menggunakan implementasi memori rendah dari pesan relasi yang melewati fungsi di pengeluaran kecepatan. Jenis:`bool`.

- **lr** – Tingkat pembelajaran. Jenis:`float`.

  Ini harus ditetapkan sebagai hyperparameter Tier-1.

- **neg-share** – Dalam prediksi link, menunjukkan apakah edge bersampel positif dapat berbagi sampel edge negatif. Jenis:`bool`.
- **num-bases** – Jumlah basis untuk dekomposisi dasar dalam model `rgcn`. Menggunakan nilai `num-bases` yang kurang dari jumlah jenis tepi dalam grafik bertindak sebagai regularizer untuk model. `rgcn` Jenis:`int`.
- **num-epochs** – Banyaknya jangka waktu pelatihan untuk dijalankan. Jenis:`int`.

  Jangka waktu adalah pelatihan lengkap yang melewati grafik.

- **num-hidden** – Ukuran lapisan tersembunyi atau jumlah unit. Jenis:`int`.

Ini juga menetapkan ukuran penanaman awal untuk simpul tanpa fitur.

Menyetel ini ke nilai yang jauh lebih besar tanpa mengurangi `batch-size` dapat menyebabkan out-of-memory masalah untuk pelatihan tentang instance GPU.

- **`num-layer`** – Jumlah layer GNN pada model. Jenis:`int`.

  Nilai ini dipasangkan erat dengan parameter fanout dan harus ada setelah fanout diatur di tier hyperparameter yang sama.

  Karena ini dapat menyebabkan kinerja model bervariasi luas, seharusnya hyperparameter ini diperbaiki atau ditetapkan sebagai hyperparameter Tier-2 atau Tier-3.
- **`num-negs`** – Dalam prediksi tautan, jumlah sampel negatif per sampel positif. Jenis:`int`.
- **`per-feat-name-embed`**— Menunjukkan apakah akan menyematkan setiap fitur dengan mengubahnya secara independen sebelum menggabungkan fitur. Jenis:`bool`.

  Ketika diatur k`true`, setiap fitur per node ditransformasikan secara independen ke ukuran dimensi tetap sebelum semua fitur yang diubah untuk node digabungkan dan selanjutnya diubah ke dimensi. `num_hidden`

  Saat disetel k`false`, fitur digabungkan tanpa transformasi khusus fitur apa pun.
- **`regularization-coef`** – Dalam prediksi link, koefisien kerugian regularisasi. Jenis:`float`.
- **`rel-part`** – Mengindikasikan apakah akan menggunakan partisi relasi untuk prediksi link `KGE`. Jenis:`bool`.
- **`sparse-lr`**— Tingkat pembelajaran untuk penyematan simpul yang dapat dipelajari. Jenis:`float`.

  Penyematan simpul awal yang dapat dipelajari digunakan untuk node tanpa fitur atau kapan disetel. `concat-node-embed` Parameter lapisan penyematan simpul yang dapat dipelajari jarang dilatih menggunakan pengoptimal terpisah yang dapat memiliki tingkat pembelajaran terpisah.
- **`use-class-weight`**— Menunjukkan apakah akan menerapkan bobot kelas untuk tugas klasifikasi yang tidak seimbang. Jika disetel k`true`, jumlah label digunakan untuk menetapkan bobot untuk setiap label kelas. Jenis:`bool`.
- **`use-edge-features`**— Menunjukkan apakah akan menggunakan fitur tepi selama pengiriman pesan. Jika disetel k`true`, modul fitur tepi khusus ditambahkan ke lapisan RGCN untuk tipe tepi yang memiliki fitur. Jenis:`bool`.
- **`use-self-loop`** – Menunjukkan apakah akan menyertakan self loop dalam pelatihan model `rgcn`. Jenis:`bool`.

- **`window-for-early-stop`**— Mengontrol jumlah skor validasi terbaru ke rata-rata untuk memutuskan pemberhentian awal. Defaultnya adalah 3. type=int. Lihat juga Penghentian awal proses pelatihan model di Neptunus ML. Jenis:`int`. Default: `3`.

  Lihat .

Menyesuaikan hiperparameter di Neptunus ML

Saat Anda mengedit file `model-HPO-configuration.json`, berikut ini adalah jenis perubahan yang paling umum dibuat:

- Edit nilai and/or maksimum minimum `range` hyperparameters.
- Menetapkan hyperparameter ke nilai tetap dengan memindahkannya ke bagian `fixed-param` dan menetapkan nilai defaultnya ke nilai tetap yang ingin Anda ambil.
- Mengubah prioritas hyperparameter dengan menempatkannya di tier tertentu, mengedit rentangnya, dan memastikan bahwa nilai defaultnya diatur dengan tepat.

Praktik terbaik pelatihan model

Ada beberapa hal yang dapat Anda lakukan untuk meningkatkan performa Neptune ML.

Pilih properti simpul yang tepat

Tidak semua properti dalam grafik Anda bermakna atau relevan dengan tugas machine learning Anda. Setiap properti yang tidak relevan harus dikecualikan selama ekspor data.

Berikut ini adalah beberapa praktik terbaik:

- Gunakan ahli domain untuk membantu mengevaluasi pentingnya fitur dan kelayakan menggunakannya untuk prediksi.
- Menghapus fitur yang Anda tentukan bersifat berlebihan atau tidak relevan untuk mengurangi kebisingan dalam data dan korelasi tidak penting.
- Iterate saat Anda membangun model Anda. Sesuaikan fitur, kombinasi fitur, dan tujuan penyetelan saat Anda mengikuti.

Pemrosesan Fiturdi Panduan Developer Amazon Machine Learning menyediakan pedoman tambahan untuk pemrosesan fitur yang relevan dengan Neptune ML.

Menangani titik data outlier

Outlier adalah titik data yang berbeda secara signifikan dari data yang tersisa. Pencilan data dapat merusak atau menyesatkan proses pelatihan, menghasilkan waktu pelatihan yang lebih lama atau model yang kurang akurat. Kecuali outlier benar-benar penting, Anda harus menghilangkan outlier sebelum mengekspor data.

Hapus simpul dan edge duplikat

Grafik yang disimpan di Neptune mungkin memiliki simpul atau edge duplikat . Elemen yang berlebihan ini akan memperkenalkan kebisingan untuk pelatihan Model ML. Hilangkan simpul atau edge duplikat sebelum mengekspor data.

Menyetel struktur grafik

Ketika grafik diekspor, Anda dapat mengubah cara fitur diproses dan cara grafik dibangun, untuk meningkatkan performa model.

Berikut ini adalah beberapa praktik terbaik:

- Ketika properti edge memiliki arti kategori edge, ada baiknya mengubahnya menjadi jenis edge dalam beberapa kasus.
- Kebijakan normalisasi default yang digunakan untuk properti numerik adalah `min-max`, tetapi dalam beberapa kasus kebijakan normalisasi lainnya bekerja lebih baik. Anda dapat melakukan pra-proses properti dan mengubah kebijakan normalisasi seperti yang dijelaskan dalam Elemen file `model-HPO-configuration.json`.
- Proses ekspor secara otomatis menghasilkan jenis fitur berdasarkan jenis properti. Sebagai contoh, itu memperlakukan properti `String` sebagai fitur kategoris dan properti `Float` dan `Int` sebagai fitur numerik. Jika perlu, Anda dapat mengubah jenis fitur setelah ekspor (lihat Elemen file `model-HPO-configuration.json`).

Menyetel rentang dan default hyperparameter

Operasi pemrosesan data menyimpulkan rentang konfigurasi hyperparameter dari grafik. Jika rentang dan default hyperparameter model yang dihasilkan tidak bekerja dengan baik untuk data grafik Anda, Anda dapat mengedit file konfigurasi HPO untuk membuat strategi penyetelan hyperparameter Anda sendiri.

Berikut ini adalah beberapa praktik terbaik:

- Ketika grafik menjadi besar, ukuran dimensi tersembunyi default mungkin tidak cukup besar untuk berisi semua informasi. Anda dapat mengubah hyperparameter `num-hidden` untuk mengontrol ukuran dimensi tersembunyi.
- Untuk model knowledge graph embedding (KGE), Anda mungkin ingin mengubah model spesifik yang digunakan sesuai dengan struktur grafik dan anggaran Anda.

  `TrainsE`model mengalami kesulitan dalam menangani hubungan one-to-many (1-N), many-to-one (N-1), dan many-to-many (N-N). `DistMult`model mengalami kesulitan dalam berurusan dengan hubungan simetris. `RotatE`pandai memodelkan semua jenis hubungan tetapi lebih mahal daripada `TrainsE` dan `DistMult` selama pelatihan.
- Dalam beberapa kasus, ketika identifikasi simpul dan informasi fitur simpul penting, Anda harus menggunakan `` `concat-node-embed` `` untuk memberitahu model Neptune ML untuk mendapatkan representasi awal dari simpul dengan menggabungkan fitur-fiturnya dengan penanaman awalnya.
- Ketika Anda mendapatkan performa yang cukup baik atas beberapa hyperparameters, Anda dapat menyesuaikan ruang pencarian hyperparameter sesuai dengan hasil tersebut.

Penghentian awal proses pelatihan model di Neptunus ML

Penghentian dini dapat secara signifikan mengurangi waktu lari pelatihan model dan biaya terkait tanpa menurunkan kinerja model. Ini juga mencegah model agar tidak terlalu pas pada data pelatihan.

Penghentian dini tergantung pada pengukuran reguler kinerja set validasi. Awalnya, kinerja meningkat saat pelatihan berlangsung, tetapi ketika model mulai overfitting, itu mulai menurun lagi. Fitur penghentian awal mengidentifikasi titik di mana model mulai overfitting dan menghentikan pelatihan model pada saat itu.

Neptunus ML memantau panggilan metrik validasi dan membandingkan metrik validasi terbaru dengan rata-rata metrik validasi selama evaluasi **n**terakhir, di mana angka ditetapkan menggunakan parameter. **n**`window-for-early-stop` Segera setelah metrik validasi lebih buruk dari rata-rata itu, Neptunus ML menghentikan pelatihan model dan menyimpan model terbaik sejauh ini.

Anda dapat mengontrol penghentian awal menggunakan parameter berikut:

- **`window-for-early-stop`**— Nilai parameter ini adalah bilangan bulat yang menentukan jumlah skor validasi terbaru menjadi rata-rata saat memutuskan pemberhentian awal. Nilai default-nya adalah `3`.
- **`enable-early-stop`**— Gunakan parameter Boolean ini untuk mematikan fitur berhenti awal. Secara default, nilainya adalah`true`.

Penghentian awal proses HPO di Neptunus ML

Fitur berhenti awal di Neptunus ML juga menghentikan pekerjaan pelatihan yang tidak berkinerja baik dibandingkan dengan pekerjaan pelatihan lainnya, menggunakan fitur AI HPO SageMaker warm-start. Hal ini juga dapat mengurangi biaya dan meningkatkan kualitas HPO.

Lihat Menjalankan pekerjaan tuning hyperparameter start yang hangat untuk deskripsi cara kerjanya.

Warm start memberikan kemampuan untuk menyampaikan informasi yang dipelajari dari pekerjaan pelatihan sebelumnya ke pekerjaan pelatihan berikutnya dan memberikan dua manfaat berbeda:

- Pertama, hasil pekerjaan pelatihan sebelumnya digunakan untuk memilih kombinasi hiperparameter yang baik untuk dicari dalam pekerjaan penyetelan baru.
- Kedua, memungkinkan penghentian awal untuk mengakses lebih banyak model berjalan, yang mengurangi waktu penyetelan.

Fitur ini diaktifkan secara otomatis di Neptunus ML, dan memungkinkan Anda mencapai keseimbangan antara waktu pelatihan model dan kinerja. Jika Anda puas dengan kinerja model saat ini, Anda dapat menggunakan model itu. Jika tidak, Anda menjalankan lebih banyak HPOs yang dimulai dengan hasil lari sebelumnya untuk menemukan model yang lebih baik.

Dapatkan layanan dukungan profesional

AWS menawarkan layanan dukungan profesional untuk membantu Anda dengan masalah dalam pembelajaran mesin Anda pada proyek Neptunus. Jika Anda terjebak, hubungi dukungan AWS.

Gunakan model terlatih untuk menghasilkan artefak model baru

Dengan menggunakan perintah transformasi model Neptunus ML, Anda dapat menghitung artefak model seperti penyematan simpul pada data grafik yang diproses menggunakan parameter model yang telah dilatih sebelumnya.

Transformasi model untuk inferensi inkremental

Dalam alur kerja inferensi model inkremental, setelah Anda memproses data grafik terbaru yang Anda ekspor dari Neptunus, Anda dapat memulai pekerjaan transformasi model menggunakan perintah curl (atau awscurl) seperti berikut:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "mlModelTrainingJobId": "(the ML model training job-id)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
      }'
```

Anda kemudian dapat meneruskan ID pekerjaan ini ke panggilan API create-endpoint untuk membuat titik akhir baru atau memperbarui yang sudah ada dengan artefak model baru yang dihasilkan oleh pekerjaan ini. Ini memungkinkan titik akhir baru atau yang diperbarui untuk memberikan prediksi model untuk data grafik yang diperbarui.

Transformasi model untuk pekerjaan pelatihan apa pun

Anda juga dapat menyediakan `trainingJobName` parameter untuk menghasilkan artefak model untuk salah satu pekerjaan pelatihan SageMaker AI yang diluncurkan selama pelatihan model Neptunus ML. Karena pekerjaan pelatihan model Neptunus ML berpotensi meluncurkan SageMaker banyak pekerjaan pelatihan AI, ini memberi Anda fleksibilitas untuk membuat titik akhir inferensi berdasarkan salah satu pekerjaan pelatihan AI tersebut. SageMaker

Misalnya:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
```

```
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "trainingJobName" : "(name a completed SageMaker AI training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
      }'
```

Jika pekerjaan pelatihan asli adalah untuk model kustom yang disediakan pengguna, Anda harus menyertakan `customModelTransformParameters` objek saat menjalankan transformasi model. Lihat Model khusus di Neptunus ML untuk informasi tentang cara menerapkan dan menggunakan model kustom.

> **Note**
>
> `modeltransform`Perintah selalu menjalankan transformasi model pada pekerjaan pelatihan SageMaker AI terbaik untuk pelatihan itu.

Lihat Perintah modeltransform untuk informasi lebih lanjut tentang pekerjaan transformasi model.

Artefak yang diproduksi oleh pelatihan model di Neptunus ML

Setelah pelatihan model, Neptunus ML menggunakan parameter model terlatih terbaik untuk menghasilkan artefak model yang diperlukan untuk meluncurkan titik akhir inferensi dan memberikan prediksi model. Artefak ini dikemas oleh pekerjaan pelatihan dan disimpan di lokasi keluaran Amazon S3 dari pekerjaan pelatihan AI SageMaker terbaik.

Bagian berikut menjelaskan apa yang termasuk dalam artefak model untuk berbagai tugas, dan bagaimana perintah transformasi model menggunakan model terlatih yang sudah ada sebelumnya untuk menghasilkan artefak bahkan pada data grafik baru.

Artefak yang dihasilkan untuk tugas yang berbeda

Isi artefak model yang dihasilkan oleh proses pelatihan tergantung pada tugas pembelajaran mesin target:

- Klasifikasi dan regresi node — Untuk prediksi properti node, artefak mencakup parameter model, penyematan simpul dari encoder GNN, prediksi model untuk node dalam grafik pelatihan, dan beberapa file konfigurasi untuk titik akhir inferensi. Dalam klasifikasi node dan tugas regresi node, prediksi model dihitung sebelumnya untuk node yang ada selama pelatihan untuk mengurangi latensi kueri.
- Klasifikasi dan regresi tepi — Untuk prediksi properti tepi, artefak juga menyertakan parameter model dan penyematan simpul. Parameter dekoder model sangat penting untuk inferensi karena kami menghitung klasifikasi tepi atau prediksi regresi tepi dengan menerapkan decoder model ke penyematan simpul sumber dan tujuan tepi.
- Prediksi tautan - Untuk prediksi tautan, selain artefak yang dihasilkan untuk prediksi properti tepi, grafik DGL juga disertakan sebagai artefak karena prediksi tautan memerlukan grafik pelatihan untuk melakukan prediksi. Tujuan prediksi tautan adalah untuk memprediksi simpul tujuan yang cenderung bergabung dengan simpul sumber untuk membentuk tepi jenis tertentu dalam grafik. Untuk melakukan ini, penyematan simpul dari simpul sumber dan representasi yang dipelajari untuk tipe tepi digabungkan dengan penyematan simpul dari semua simpul tujuan yang mungkin untuk menghasilkan skor kemungkinan tepi untuk masing-masing simpul tujuan. Skor kemudian diurutkan untuk menentukan peringkat simpul tujuan potensial dan mengembalikan kandidat teratas.

Untuk setiap jenis tugas, bobot model Graph Neural Network dari DGL disimpan dalam artefak model. Hal ini memungkinkan Neptunus ML untuk menghitung output model baru saat grafik berubah

(inferensi induktif), selain menggunakan prediksi dan penyematan yang telah dihitung sebelumnya (inferensi transduktif) untuk mengurangi latensi.

Menghasilkan artefak model baru

Artefak model yang dihasilkan setelah pelatihan model di Neptunus ML terkait langsung dengan proses pelatihan. Ini berarti bahwa penyematan dan prediksi yang telah dihitung sebelumnya hanya ada untuk entitas yang ada dalam grafik pelatihan asli. Meskipun mode inferensi induktif untuk titik akhir Neptunus ML dapat menghitung prediksi untuk entitas baru secara real-time, Anda mungkin ingin menghasilkan prediksi batch pada entitas baru tanpa menanyakan titik akhir.

Untuk mendapatkan prediksi model batch untuk entitas baru yang telah ditambahkan ke grafik, artefak model baru perlu dihitung ulang untuk data grafik baru. Ini dilakukan dengan menggunakan `modeltransform` perintah. Anda menggunakan `modeltransform` perintah ketika Anda hanya ingin prediksi batch tanpa menyiapkan titik akhir, atau ketika Anda ingin semua prediksi dihasilkan sehingga Anda dapat menulisnya kembali ke grafik.

Karena pelatihan model secara implisit melakukan transformasi model pada akhir proses pelatihan, artefak model selalu dihitung ulang pada data grafik pelatihan oleh pekerjaan pelatihan. Namun, `modeltransform` perintah tersebut juga dapat menghitung artefak model pada data grafik yang tidak digunakan untuk melatih model. Untuk ini, data grafik baru harus diproses menggunakan pengkodean fitur yang sama dengan data grafik asli dan harus mematuhi skema grafik yang sama.

Anda dapat melakukannya dengan terlebih dahulu membuat pekerjaan pemrosesan data baru yang merupakan tiruan dari pekerjaan pemrosesan data yang dijalankan pada data grafik pelatihan asli, dan menjalankannya pada data grafik baru (lihatMemproses data grafik yang diperbarui untuk Neptunus ML). Kemudian, panggil `modeltransform` perintah dengan yang baru `dataProcessingJobId` dan yang lama `modelTrainingJobId` untuk menghitung ulang artefak model pada data grafik yang diperbarui.

Untuk prediksi properti node, penyematan dan prediksi simpul dihitung ulang pada data grafik baru, bahkan untuk node yang ada dalam grafik pelatihan asli.

Untuk prediksi properti edge dan prediksi tautan, penyematan simpul juga dihitung ulang dan juga mengesampingkan penyematan simpul yang ada. Untuk menghitung ulang penyematan simpul, Neptunus MLmenerapkan encoder GNN yang dipelajari dari model terlatih sebelumnya ke node data grafik baru dengan fitur baru mereka.

Untuk node yang tidak memiliki fitur, representasi awal yang dipelajari dari pelatihan model asli digunakan kembali. Untuk node baru yang tidak memiliki fitur dan tidak ada dalam grafik pelatihan

asli, Neptunus ML menginisialisasi representasi mereka sebagai rata-rata representasi simpul awal yang dipelajari dari tipe simpul yang ada dalam grafik pelatihan asli. Ini dapat menyebabkan beberapa penurunan kinerja dalam prediksi model jika Anda memiliki banyak node baru yang tidak memiliki fitur, karena semuanya akan diinisialisasi ke penyematan awal rata-rata untuk jenis node tersebut.

Jika model Anda dilatih dengan `concat-node-embed` set ke true, maka representasi node awal dibuat dengan menggabungkan fitur node dengan representasi awal yang dapat dipelajari. Jadi, untuk grafik yang diperbarui, representasi simpul awal dari node baru juga menggunakan penyematan simpul awal rata-rata, digabungkan dengan fitur node baru.

Selain itu, penghapusan node saat ini tidak didukung. Jika node telah dihapus dalam grafik yang diperbarui, Anda harus melatih kembali model pada data grafik yang diperbarui.

Menghitung ulang artefak model menggunakan kembali parameter model yang dipelajari pada grafik baru, dan hanya boleh dilakukan ketika grafik baru sangat mirip dengan grafik lama. Jika grafik baru Anda tidak cukup mirip, Anda perlu melatih ulang model untuk mendapatkan kinerja model yang serupa pada data grafik baru. Apa yang cukup mirip tergantung pada struktur data grafik Anda, tetapi sebagai aturan praktis Anda harus melatih kembali model Anda jika data baru Anda lebih dari 10-20% berbeda dari data grafik pelatihan asli.

Untuk grafik di mana semua node memiliki fitur, ujung ambang batas yang lebih tinggi (20% berbeda) berlaku tetapi untuk grafik di mana banyak node tidak memiliki fitur dan node baru yang ditambahkan ke grafik tidak memiliki properti, maka ujung bawah (10% berbeda) bahkan mungkin terlalu tinggi.

Lihat Perintah modeltransform untuk informasi lebih lanjut tentang pekerjaan transformasi model.

Model khusus di Neptunus ML

> **Note**
>
> Dukungan model kustom Neptunus ML bergantung pada versi Python 3 yang lebih lama. Untuk membuat dan menjalankan model GNN kustom dengan up-to-date dependensi gunakan pada. GraphStorm SageMaker
> Inferensi induktif waktu nyata saat ini tidak didukung untuk model khusus.

Neptunus ML memungkinkan Anda menentukan implementasi model kustom Anda sendiri menggunakan Python. Anda dapat melatih dan menerapkan model kustom menggunakan infrastruktur Neptunus ML seperti yang Anda lakukan untuk model bawaan, dan menggunakannya untuk mendapatkan prediksi melalui kueri grafik.

Anda dapat mulai menerapkan model kustom Anda sendiri dengan Python dengan mengikuti contoh toolkit Neptunus ML, dan dengan menggunakan komponen model yang disediakan di toolkit Neptunus ML. Bagian berikut memberikan rincian lebih lanjut.

Daftar Isi

- Ikhtisar model khusus di Neptunus ML
  - Kapan menggunakan model kustom di Neptunus ML
  - Alur kerja untuk mengembangkan dan menggunakan model kustom di Neptunus ML
- Pengembangan model khusus di Neptunus ML
  - Pengembangan skrip pelatihan model khusus di Neptunus ML
  - Model kustom mengubah pengembangan skrip di Neptunus ML
  - model-hpo-configuration.jsonFile kustom di Neptunus ML
  - Pengujian lokal implementasi model kustom Anda di Neptunus ML

Ikhtisar model khusus di Neptunus ML

Kapan menggunakan model kustom di Neptunus ML

Model bawaan Neptunus ML menangani semua tugas standar yang didukung oleh Neptunus ML, tetapi mungkin ada kasus di mana Anda ingin memiliki kontrol yang lebih terperinci atas model untuk tugas tertentu, atau perlu menyesuaikan proses pelatihan model. Misalnya, model khusus sesuai dalam situasi berikut:

- Pengkodean fitur untuk fitur teks dari model teks yang sangat besar perlu dijalankan pada GPU.
- Anda ingin menggunakan model Graph Neural Network (GNN) kustom Anda sendiri yang dikembangkan di Deep Graph Library (DGL).
- Anda ingin menggunakan model tabel atau model ansambel untuk klasifikasi dan regresi simpul.

Alur kerja untuk mengembangkan dan menggunakan model kustom di Neptunus ML

Dukungan model khusus di Neptunus ML dirancang untuk diintegrasikan dengan mulus ke dalam alur kerja Neptunus ML yang ada. Ia bekerja dengan menjalankan kode kustom dalam modul sumber Anda pada infrastruktur Neptunus ML untuk melatih model. Sama seperti halnya untuk mode bawaan, Neptunus MLsecara otomatis meluncurkan SageMaker pekerjaan penyetelan HyperParameter AI dan memilih model terbaik sesuai dengan metrik evaluasi. Kemudian menggunakan implementasi yang disediakan dalam modul sumber Anda untuk menghasilkan artefak model untuk penerapan.

Ekspor data, konfigurasi pelatihan, dan preprocessing data sama untuk model kustom seperti model bawaan.

Setelah preprocessing data adalah ketika Anda dapat secara iteratif dan interaktif mengembangkan dan menguji implementasi model kustom Anda menggunakan Python. Ketika model Anda siap produksi, Anda dapat mengunggah modul Python yang dihasilkan ke Amazon S3 seperti ini:

```
aws s3 cp --recursive (source path to module) s3://(bucket name)/(destination path for
 your module)
```

Kemudian, Anda dapat menggunakan default normal atau alur kerja data tambahan untuk menyebarkan model ke produksi, dengan beberapa perbedaan.

Untuk pelatihan model menggunakan model kustom, Anda harus menyediakan objek `customModelTrainingParameters` JSON ke API pelatihan model

Neptunus ML untuk memastikan bahwa kode kustom Anda digunakan. Bidang dalam `customModelTrainingParameters` objek adalah sebagai berikut:

- **`sourceS3DirectoryPath`**— (Wajib) Jalur ke lokasi Amazon S3 tempat modul Python yang mengimplementasikan model Anda berada. Ini harus menunjuk ke lokasi Amazon S3 yang valid yang berisi, setidaknya, skrip pelatihan, skrip transformasi, dan file. `model-hpo-configuration.json`
- **`trainingEntryPointScript`**— (Opsional) Nama titik masuk dalam modul skrip Anda yang melakukan pelatihan model dan mengambil hyperparameters sebagai argumen baris perintah, termasuk hyperparameters tetap.

  Default: `training.py`.
- **`transformEntryPointScript`**— (Opsional) Nama titik masuk dalam modul skrip Anda yang harus dijalankan setelah model terbaik dari pencarian hyperparameter telah diidentifikasi, untuk menghitung artefak model yang diperlukan untuk penerapan model. Itu harus dapat berjalan tanpa argumen baris perintah.

  Default: `transform.py`.

Contoh:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Demikian pula, untuk mengaktifkan transformasi model kustom, Anda harus menyediakan objek `customModelTransformParameters` JSON ke API transformasi model Neptunus ML, dengan nilai bidang yang kompatibel dengan parameter model yang disimpan dari tugas pelatihan. `customModelTransformParameters`Objek berisi bidang-bidang ini:

- **`sourceS3DirectoryPath`**— (Wajib) Jalur ke lokasi Amazon S3 tempat modul Python yang mengimplementasikan model Anda berada. Ini harus menunjuk ke lokasi Amazon S3 yang valid yang berisi, setidaknya, skrip pelatihan, skrip transformasi, dan file. `model-hpo-configuration.json`
- **`transformEntryPointScript`**— (Opsional) Nama titik masuk dalam modul skrip Anda yang harus dijalankan setelah model terbaik dari pencarian hyperparameter telah diidentifikasi, untuk menghitung artefak model yang diperlukan untuk penerapan model. Itu harus dapat berjalan tanpa argumen baris perintah.

  Default: `transform.py`.

Contoh:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "trainingJobName" : "(name of a completed SageMaker AI training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
        "customModelTransformParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Pengembangan model khusus di Neptunus ML

Cara yang baik untuk memulai pengembangan model kustom adalah dengan mengikuti contoh toolkit Neptunus ML untuk menyusun dan menulis modul pelatihan Anda. Toolkit Neptunus ML juga mengimplementasikan komponen model HTML grafik termodulasi di modelzoo yang dapat Anda tumpuk dan gunakan untuk membuat model kustom Anda.

Selain itu, toolkit menyediakan fungsi utilitas yang membantu Anda menghasilkan artefak yang diperlukan selama pelatihan model dan transformasi model. Anda dapat mengimpor paket Python ini dalam implementasi kustom Anda. Setiap fungsi atau modul yang disediakan dalam toolkit juga tersedia di lingkungan pelatihan Neptunus ML.

Jika modul Python Anda memiliki dependensi eksternal tambahan, Anda dapat menyertakan dependensi tambahan ini dengan membuat `requirements.txt` file di direktori modul Anda. Paket yang tercantum dalam `requirements.txt` file kemudian akan diinstal sebelum skrip pelatihan Anda dijalankan.

Minimal, modul Python yang mengimplementasikan model kustom Anda harus berisi yang berikut:

- Titik masuk skrip pelatihan
- Titik masuk skrip transformasi
- Sebuah `model-hpo-configuration.json` file

Pengembangan skrip pelatihan model khusus di Neptunus ML

Skrip pelatihan model kustom Anda harus berupa skrip Python yang dapat dieksekusi seperti contoh toolkit Neptunus ML. `train.py` Itu harus menerima nama dan nilai hyperparameter sebagai argumen baris perintah. Selama pelatihan model, nama hyperparameter diperoleh dari `model-hpo-configuration.json` file. Nilai hyperparameter termasuk dalam rentang hyperparameter yang valid jika hyperparameter dapat disetel, atau mengambil nilai hyperparameter default jika tidak dapat disetel.

Skrip pelatihan Anda dijalankan pada instance pelatihan SageMaker AI menggunakan sintaks seperti ini:

```
python3 (script entry point) --(1st parameter) (1st value) --(2nd parameter) (2nd
 value) (...)
```

Untuk semua tugas, Neptunus AutoTrainer ML mengirimkan beberapa parameter yang diperlukan ke skrip pelatihan Anda selain hiperparameter yang Anda tentukan, dan skrip Anda harus dapat menangani parameter tambahan ini agar berfungsi dengan baik.

Parameter tambahan yang diperlukan ini agak berbeda menurut tugas:

Untuk klasifikasi simpul atau regresi simpul

- **`task`**— Jenis tugas yang digunakan secara internal oleh Neptunus ML. Untuk klasifikasi node ininode_class, dan untuk regresi simpul itu. node_regression
- **`model`**— Nama model yang digunakan secara internal oleh Neptunus ML, yang custom dalam hal ini.
- **`name`**— Nama tugas yang digunakan secara internal oleh Neptunus ML, yang node_class-custom untuk klasifikasi simpul dalam kasus ini, dan untuk regresi simpul. node_regression-custom
- **`target_ntype`**— Nama tipe node untuk klasifikasi atau regresi.
- **`property`**— Nama properti node untuk klasifikasi atau regresi.

Untuk prediksi tautan

- **`task`**— Jenis tugas yang digunakan secara internal oleh Neptunus ML. Untuk prediksi tautan, inilink_predict.
- **`model`**— Nama model yang digunakan secara internal oleh Neptunus ML, yang custom dalam hal ini.
- **`name`**— Nama tugas yang digunakan secara internal oleh Neptunus ML, yang link_predict-custom dalam kasus ini.

Untuk klasifikasi tepi atau regresi tepi

- **`task`**— Jenis tugas yang digunakan secara internal oleh Neptunus ML. Untuk klasifikasi tepi iniedge_class, dan untuk regresi tepi memang demikian. edge_regression
- **`model`**— Nama model yang digunakan secara internal oleh Neptunus ML, yang custom dalam hal ini.
- **`name`**— Nama tugas yang digunakan secara internal oleh Neptunus ML, yang edge_class-custom untuk klasifikasi tepi dalam kasus ini, edge_regression-custom dan untuk regresi tepi.
- **`target_etype`**— Nama tipe tepi untuk klasifikasi atau regresi.

- **`property`**— Nama properti tepi untuk klasifikasi atau regresi.

Script Anda harus menyimpan parameter model, serta artefak lain yang akan diperlukan untuk pada akhir pelatihan.

Anda dapat menggunakan fungsi utilitas toolkit Neptunus ML untuk menentukan lokasi data grafik yang diproses, lokasi di mana parameter model harus disimpan, dan perangkat GPU apa yang tersedia pada instance pelatihan. Lihat contoh skrip pelatihan train.py untuk contoh cara menggunakan fungsi utilitas ini.

Model kustom mengubah pengembangan skrip di Neptunus ML

Skrip transformasi diperlukan untuk memanfaatkan alur kerja inkremental Neptunus ML untuk inferensi model pada grafik yang berkembang tanpa melatih ulang model. Bahkan jika semua artefak yang diperlukan untuk penerapan model dihasilkan oleh skrip pelatihan, Anda masih perlu menyediakan skrip transformasi jika Anda ingin menghasilkan model yang diperbarui tanpa melatih ulang model.

> **Note**
>
> Inferensi induktif waktu nyata saat ini tidak didukung untuk model khusus.

Skrip transformasi model kustom Anda harus berupa skrip Python yang dapat dieksekusi seperti skrip contoh transform.py toolkit Neptunus MLkit. Karena skrip ini dipanggil selama pelatihan model tanpa argumen baris perintah, argumen baris perintah apa pun yang diterima skrip harus memiliki default.

Skrip berjalan pada instance pelatihan SageMaker AI dengan sintaks seperti ini:

```
python3 (your transform script entry point)
```

Skrip transformasi Anda akan membutuhkan berbagai informasi, seperti:

- Lokasi data grafik yang diproses.
- Lokasi di mana parameter model disimpan dan di mana artefak model baru harus disimpan.
- Perangkat yang tersedia pada instance.
- Hyperparameter yang menghasilkan model terbaik.

Input ini diperoleh dengan menggunakan fungsi utilitas Neptunus ML yang dapat dipanggil skrip Anda. Lihat contoh skrip transform.py toolkit untuk contoh cara melakukannya.

Skrip harus menyimpan penyematan simpul, pemetaan ID simpul, dan artefak lain yang diperlukan untuk penerapan model untuk setiap tugas. Lihat dokumentasi artefak model untuk informasi lebih lanjut tentang artefak model yang diperlukan untuk tugas ML Neptunus yang berbeda.

`model-hpo-configuration.json`File kustom di Neptunus ML

`model-hpo-configuration.json`File mendefinisikan hyperparameters untuk model kustom Anda. Ini dalam format yang sama dengan `model-hpo-configuration.json` file yang digunakan dengan model bawaan Neptunus ML, dan lebih diutamakan daripada versi yang dibuat secara otomatis oleh Neptunus ML dan diunggah ke lokasi data yang Anda proses.

Ketika Anda menambahkan hyperparameter baru ke model Anda, Anda juga harus menambahkan entri untuk hyperparameter dalam file ini sehingga hyperparameter diteruskan ke skrip pelatihan Anda.

Anda harus menyediakan rentang untuk hyperparameter jika Anda ingin dapat disetel, dan mengaturnya sebagai, `tier-1tier-2`, atau `tier-3` param. Hyperparameter akan disetel jika jumlah total pekerjaan pelatihan yang dikonfigurasi memungkinkan untuk menyetel hiperparameter di tingkatnya. Untuk parameter yang tidak dapat disetel, Anda harus memberikan nilai default dan menambahkan hyperparameter ke `fixed-param` bagian file. Lihat contoh `model-hpo-configuration.json`file contoh toolkit untuk contoh cara melakukannya.

Anda juga harus memberikan definisi metrik yang akan digunakan pekerjaan HyperParameter Optimasi SageMaker AI untuk mengevaluasi model kandidat yang dilatih. Untuk melakukan ini, Anda menambahkan objek `eval_metric` JSON ke `model-hpo-configuration.json` file seperti ini:

```
"eval_metric": {
  "tuning_objective": {
      "MetricName": "(metric_name)",
      "Type": "Maximize"
  },
  "metric_definitions": [
    {
      "Name": "(metric_name)",
      "Regex": "(metric regular expression)"
    }
  ]
```

```
},
```

`metric_definitions`Array dalam `eval_metric` objek mencantumkan objek definisi metrik untuk setiap metrik yang Anda ingin SageMaker AI ekstrak dari instance pelatihan. Setiap objek definisi metrik memiliki `Name` kunci yang memungkinkan Anda memberikan nama untuk metrik (seperti “akurasi”, “f1", dan seterusnya) `Regex` Kunci ini memungkinkan Anda memberikan string ekspresi reguler yang cocok dengan cara metrik tertentu dicetak dalam log pelatihan. Lihat halaman SageMaker AI HyperParameter Tuning untuk detail selengkapnya tentang cara menentukan metrik.

`tuning_objective`Objek `eval_metric` kemudian memungkinkan Anda untuk menentukan metrik mana yang `metric_definitions` harus digunakan sebagai metrik evaluasi yang berfungsi sebagai metrik objektif untuk optimasi hyperparameter. Nilai untuk `MetricName` harus sesuai dengan nilai a `Name` dalam salah satu definisi di`metric_definitions`. Nilai untuk `Type` harus berupa “Maksimalkan” atau “Minimalkan” tergantung pada apakah metrik harus ditafsirkan sebagai greater-is-better (seperti “akurasi”) atau less-is-better (seperti "mean-squared-error”.

Kesalahan di bagian `model-hpo-configuration.json` file ini dapat mengakibatkan kegagalan pekerjaan API pelatihan model Neptunus ML, karena SageMaker pekerjaan Tuning HyperParameter AI tidak akan dapat memilih model terbaik.

Pengujian lokal implementasi model kustom Anda di Neptunus ML

Anda dapat menggunakan lingkungan Conda toolkit Neptunus ML untuk menjalankan kode Anda secara lokal untuk menguji dan memvalidasi model Anda. Jika Anda mengembangkan instance Notebook Neptunus, maka lingkungan Conda ini akan diinstal sebelumnya pada instance Notebook Neptunus. Jika Anda mengembangkan pada instance yang berbeda, maka Anda harus mengikuti instruksi penyiapan lokal di toolkit Neptunus ML.

Lingkungan Conda secara akurat mereproduksi lingkungan tempat model Anda akan berjalan saat Anda memanggil API pelatihan model. Semua contoh skrip pelatihan dan skrip transformasi memungkinkan Anda untuk meneruskan `--local` flag baris perintah untuk menjalankan skrip di lingkungan lokal agar mudah debugging. Ini adalah praktik yang baik saat mengembangkan model Anda sendiri karena memungkinkan Anda untuk menguji implementasi model Anda secara interaktif dan berulang. Selama pelatihan model di lingkungan pelatihan produksi Neptunus ML, parameter ini dihilangkan.

Membuat titik akhir inferensi untuk kueri

Titik akhir inferensi memungkinkan Anda mengkueri satu model tertentu yang dibangun oleh proses pelatihan model. Titik akhir menempel pada model dengan performa terbaik dari jenis yang diberikan yang dihasilkan proses pelatihan. Endpoint ini kemudian dapat menerima kueri Gremlin dari Neptune dan mengembalikan prediksi model tersebut untuk input dalam kueri. Setelah Anda membuat titik akhir inferensi, itu tetap aktif sampai Anda menghapusnya.

Mengelola endpoint inferensi untuk Neptune ML

Setelah Anda menyelesaikan pelatihan model pada data yang Anda ekspor dari Neptune, Anda dapat membuat titik akhir inferensi menggunakan perintah `curl` (atau `awscurl`) seperti berikut:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Anda juga dapat membuat titik akhir inferensi dari model yang dibuat oleh pekerjaan transformasi model yang telah selesai, dengan cara yang hampir sama:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTransformJobId": "(the model-transform job-id of a completed job)"
      }'
```

Rincian tentang cara menggunakan perintah ini dijelaskan dalamPerintah titik akhir, bersama dengan informasi tentang cara mendapatkan status titik akhir, cara menghapus titik akhir, dan cara membuat daftar semua titik akhir inferensi.

Kueri inferensi di Neptunus ML

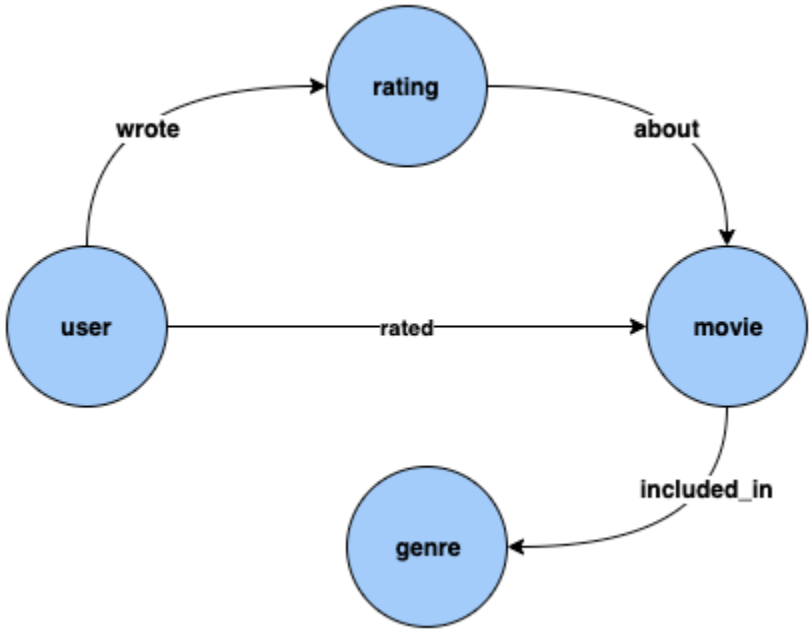
Anda dapat menggunakan Gremlin atau SPARQL untuk menanyakan titik akhir inferensi Neptunus ML. Inferensi induktif waktu nyata, bagaimanapun, saat ini hanya didukung untuk kueri Gremlin.

Kueri inferensi Gremlin di Neptune ML

Seperti dijelaskan dalamKemampuan Neptune ML, Neptunus ML mendukung model pelatihan yang dapat melakukan jenis tugas inferensi berikut:

- Klasifikasi simpul – Memprediksi fitur kategoris dari properti vertex.
- Regresi simpul — Memprediksi properti numerik dari sebuah simpul.
- Klasifikasi tepi - Memprediksi fitur kategoris properti tepi.
- Regresi tepi - Memprediksi properti numerik tepi.
- Prediksi tautan - Memprediksi node tujuan yang diberikan node sumber dan tepi keluar, atau node sumber yang diberi node tujuan dan tepi masuk.

Kami dapat mengilustrasikan tugas-tugas yang berbeda ini dengan contoh yang menggunakan kumpulan data MovieLens 100k yang disediakan oleh Research. GroupLens Dataset ini terdiri dari film, pengguna, dan peringkat film oleh pengguna, dari mana kami telah membuat grafik properti seperti ini:



Klasifikasi simpul: Dalam kumpulan data di atas, `Genre` adalah tipe simpul yang terhubung ke tipe `Movie` simpul berdasarkan tepi. `included_in` Namun, jika kita mengubah kumpulan data untuk membuat fitur kategoris untuk tipe simpul`Movie`, maka masalah menyimpulkan `Genre` untuk film baru yang ditambahkan ke grafik pengetahuan kita dapat diselesaikan dengan `Genre` menggunakan model klasifikasi simpul.

Regresi simpul: Jika kita mempertimbangkan tipe simpul`Rating`, yang memiliki properti seperti `timestamp` dan`score`, maka masalah menyimpulkan nilai numerik `Score` untuk a `Rating` dapat diselesaikan dengan menggunakan model regresi simpul.

Klasifikasi tepi: Demikian pula, untuk `Rated` tepi, jika kita memiliki properti `Scale` yang dapat memiliki salah satu nilai`Love`,`Like`,`Dislike`,`Neutral`,`Hate`,,, maka masalah menyimpulkan `Scale` untuk `Rated` tepi untuk yang baru movies/ratings dapat diselesaikan dengan menggunakan model klasifikasi tepi.

Regresi tepi: Demikian pula, untuk `Rated` tepi yang sama, jika kita memiliki properti `Score` yang memegang nilai numerik untuk peringkat, maka ini dapat disimpulkan dari model regresi tepi.

Prediksi tautan: Masalah seperti, temukan sepuluh pengguna teratas yang paling mungkin menilai film tertentu, atau menemukan sepuluh Film teratas yang kemungkinan besar akan dinilai oleh pengguna tertentu, berada di bawah prediksi tautan.

> **Note**
>
> Untuk kasus penggunaan Neptunus ML, kami memiliki seperangkat notebook yang sangat kaya yang dirancang untuk memberi Anda pemahaman langsung tentang setiap kasus penggunaan. Anda dapat membuat notebook ini bersama dengan cluster Neptunus Anda saat Anda menggunakan template Neptunus ML untuk membuat cluster Neptunus CloudFormation ML. Notebook ini juga tersedia di github.

Topik

- Predikat Neptune ML digunakan dalam query inferensi Gremlin
- Kueri klasifikasi simpul Gremlin di Neptunus ML
- Kueri regresi simpul Gremlin di Neptunus ML
- Kueri klasifikasi tepi Gremlin di Neptunus ML
- Kueri regresi tepi Gremlin di Neptunus ML
- Kueri prediksi tautan Gremlin menggunakan model prediksi tautan di Neptunus ML
- Daftar pengecualian untuk kueri inferensi Gremlin Neptune ML

Predikat Neptune ML digunakan dalam query inferensi Gremlin

`Neptune#ml.deterministic`

Predikat ini adalah opsi untuk kueri inferensi induktif — yaitu, untuk kueri yang menyertakan predikat. `Neptune#ml.inductiveInference`

Saat menggunakan inferensi induktif, mesin Neptunus membuat subgraf yang sesuai untuk mengevaluasi model GNN yang terlatih, dan persyaratan subgraf ini bergantung pada parameter model akhir. Secara khusus, `num-layer` parameter menentukan jumlah hop traversal dari node atau tepi target, dan `fanouts` parameter menentukan berapa banyak tetangga untuk sampel pada setiap hop (lihat parameter HPO).

Secara default, kueri inferensi induktif berjalan dalam mode non-deterministik, di mana Neptunus membangun lingkungan secara acak. Saat membuat prediksi, pengambilan sampel tetangga-acak yang normal ini terkadang menghasilkan prediksi yang berbeda.

Saat Anda memasukkan `Neptune#ml.deterministic` dalam kueri inferensi induktif, mesin Neptunus mencoba mengambil sampel tetangga dengan cara deterministik sehingga beberapa pemanggilan dari kueri yang sama mengembalikan hasil yang sama setiap saat. Hasilnya tidak dapat dijamin sepenuhnya deterministik, karena perubahan pada grafik dan artefak yang mendasari sistem terdistribusi masih dapat menimbulkan fluktuasi.

Anda menyertakan `Neptune#ml.deterministic` predikat dalam kueri seperti ini:

```
.with("Neptune#ml.deterministic")
```

Jika `Neptune#ml.deterministic` predikat disertakan dalam kueri yang tidak juga termasuk`Neptune#ml.inductiveInference`, itu diabaikan begitu saja.

`Neptune#ml.disableInductiveInferenceMetadataCache`

Predikat ini adalah opsi untuk kueri inferensi induktif — yaitu, untuk kueri yang menyertakan predikat. `Neptune#ml.inductiveInference`

Untuk kueri inferensi induktif, Neptunus menggunakan file metadata yang disimpan di Amazon S3 untuk menentukan jumlah hop dan fanout saat membangun lingkungan. Neptunus biasanya menyimpan metadata model ini untuk menghindari pengambilan file dari Amazon S3 berulang kali. Caching dapat dinonaktifkan dengan memasukkan `Neptune#ml.disableInductiveInferenceMetadataCache` predikat dalam kueri. Meskipun mungkin lebih lambat bagi Neptunus untuk mengambil metadata langsung dari Amazon S3, ini

SageMaker berguna ketika titik akhir AI telah diperbarui setelah pelatihan ulang atau transformasi dan cache sudah basi.

Anda menyertakan `Neptune#ml.disableInductiveInferenceMetadataCache` predikat dalam kueri seperti ini:

```
.with("Neptune#ml.disableInductiveInferenceMetadataCache")
```

Berikut adalah bagaimana kueri sampel mungkin terlihat di notebook Jupyter:

```
%%gremlin
g.with("Neptune#ml.endpoint", "ep1")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .with("Neptune#ml.disableInductiveInferenceMetadataCache")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
```

`Neptune#ml.endpoint`

Predikat `Neptune#ml.endpoint` digunakan dalam langkah `with()` untuk menentukan titik akhir inferensi, jika perlu:

```
 .with("Neptune#ml.endpoint", "the model's SageMaker AI inference endpoint")
```

Anda dapat mengidentifikasi titik akhir baik dengan `id` atau URL-nya. Contoh:

```
 .with( "Neptune#ml.endpoint", "node-classification-movie-lens-endpoint" )
```

Atau:

```
 .with( "Neptune#ml.endpoint", "https://runtime.sagemaker.us-east-1.amazonaws.com/
endpoints/node-classification-movie-lens-endpoint/invocations" )
```

> **Note**
>
> Jika Anda mengatur parameter `neptune_ml_endpoint` di grup parameter klaster Neptune DB Anda ke titik akhir `id` atau URL, Anda tidak perlu menyertakan predikat `Neptune#ml.endpoint`dalam setiap kueri.

Neptune#ml.iamRoleArn

`Neptune#ml.iamRoleArn`digunakan dalam `with()` langkah untuk menentukan ARN dari peran IAM eksekusi SageMaker AI, jika perlu:

```
.with("Neptune#ml.iamRoleArn", "the ARN for the SageMaker AI execution IAM role")
```

Untuk informasi tentang cara membuat peran IAM eksekusi SageMaker AI, lihatBuat NeptuneSageMaker IAMRole peran khusus.

> **Note**
>
> Jika Anda menyetel `neptune_ml_iam_role` parameter dalam grup parameter cluster Neptunus DB ke ARN peran IAM eksekusi AI SageMaker Anda, Anda tidak perlu menyertakan predikat di setiap kueri. `Neptune#ml.iamRoleArn`

Neptunus #ml .InductiveInference

Inferensi transduktif diaktifkan secara default di Gremlin. Untuk membuat kueri inferensi induktif real-time, sertakan `Neptune#ml.inductiveInference` predikat seperti ini:

```
.with("Neptune#ml.inductiveInference")
```

Jika grafik Anda dinamis, inferensi induktif seringkali merupakan pilihan terbaik, tetapi jika grafik Anda statis, inferensi transduktif lebih cepat dan lebih efisien.

Neptune#ml.limit

`Neptune#ml.limit`Predikat secara opsional membatasi jumlah hasil yang dikembalikan per entitas:

```
.with( "Neptune#ml.limit", 2 )
```

Secara default, batasnya adalah 1, dan jumlah maksimum yang dapat diatur adalah 100.

Neptune#ml.threshold

Predikat `Neptune#ml.threshold` secara opsional menetapkan ambang batas cutoff untuk skor hasil:

```
.with( "Neptune#ml.threshold", 0.5D )
```

Hal ini memungkinkan Anda membuang semua hasil dengan skor di bawah ambang batas yang ditentukan.

Neptune#ml.classification

`Neptune#ml.classification`Predikat dilampirkan pada `properties()` langkah untuk menetapkan bahwa properti perlu diambil dari titik akhir SageMaker AI dari model klasifikasi node:

```
.properties( "property key of the node classification
model" ).with( "Neptune#ml.classification" )
```

Neptune#ml.regression

`Neptune#ml.regression`Predikat dilampirkan pada `properties()` langkah untuk menetapkan bahwa properti perlu diambil dari titik akhir SageMaker AI dari model regresi node:

```
.properties( "property key of the node regression
model" ).with( "Neptune#ml.regression" )
```

Neptune#ml.prediction

Predikat `Neptune#ml.prediction` dilampirkan ke langkah `in()` dan `out()` untuk menetapkan bahwa ini adalah kueri prediksi link:

```
.in("edge label of the link prediction
model").with("Neptune#ml.prediction").hasLabel("target node label")
```

Neptune#ml.score

`Neptune#ml.score`Predikat digunakan dalam kueri klasifikasi simpul atau tepi Gremlin untuk mengambil Skor kepercayaan pembelajaran mesin. `Neptune#ml.score`Predikat harus diteruskan bersama dengan predikat kueri dalam `properties()` langkah untuk mendapatkan skor kepercayaan ML untuk kueri klasifikasi simpul atau tepi.

Anda dapat menemukan contoh klasifikasi simpul dengan contoh klasifikasi simpul lainnya, dan contoh klasifikasi tepi di bagian klasifikasi tepi.

Kueri klasifikasi simpul Gremlin di Neptunus ML

Untuk klasifikasi node Gremlin di Neptunus ML:

- Model ini dilatih pada satu properti dari vertex. Rangkain nilai unik dari properti ini disebut sebagai satu set kelas simpul, atau hanya, kelas.
- Kelas simpul atau nilai properti kategoris dari properti vertex ini dapat disimpulkan dari model klasifikasi simpul. Hal ini berguna di mana properti ini belum melekat pada simpul.
- Dalam rangka untuk mengambil satu kelas atau lebih dari model klasifikasi simpul, Anda perlu menggunakan langkah `with()` dengan predikat `Neptune#ml.classification` untuk mengonfigurasi langkah `properties()`. Format output mirip dengan apa yang Anda harapkan jika format adalah properti vertex.

> **Note**
>
> Klasifikasi node hanya bekerja dengan nilai properti string. Itu berarti bahwa nilai properti numerik seperti `0` atau tidak `1` didukung, meskipun string setara `"0"` dan adalah. `"1"` Demikian pula, nilai properti Boolean `true` dan `false` tidak berfungsi, tetapi `"true"` dan `"false"` lakukan.

Berikut adalah contoh kueri klasifikasi node:

```
g.with( "Neptune#ml.endpoint","node-classification-movie-lens-endpoint" )
 .with( "Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role" )
 .with( "Neptune#ml.limit", 2 )
 .with( "Neptune#ml.threshold", 0.5D )
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre").with("Neptune#ml.classification")
```

Output dari kueri ini akan terlihat seperti berikut ini:

```
==>vp[genre->Action]
==>vp[genre->Crime]
==>vp[genre->Comedy]
```

Dalam query di atas, langkah `V()` dan `properties()` digunakan sebagai berikut:

Langkah `V()` berisi rangkaian vertex di mana Anda ingin mengambil kelas dari model klasifikasi node:

```
  .V( "movie_1", "movie_2", "movie_3" )
```

`properties()`Langkah berisi kunci di mana model dilatih, dan `.with("Neptune#ml.classification")` harus menunjukkan bahwa ini adalah kueri inferensi MS klasifikasi node.

Kunci properti ganda saat ini tidak didukung di langkah `properties().with("Neptune#ml.classification")`. Sebagai contoh, hasil kueri berikut dalam pengecualian:

```
g.with("Neptune#ml.endpoint", "node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre", "other_label").with("Neptune#ml.classification")
```

Untuk pesan kesalahan tertentu, lihat daftar pengecualian Neptune ML.

Langkah `properties().with("Neptune#ml.classification")` dapat digunakan dalam kombinasi dengan semua langkah-langkah berikut:

- `value()`
- `value().is()`
- `hasValue()`
- `has(value,"")`
- `key()`
- `key().is()`
- `hasKey()`
- `has(key,"")`
- `path()`

Kueri klasifikasi simpul lainnya

Jika titik akhir inferensi dan peran IAM yang sesuai telah disimpan dalam grup parameter cluster DB Anda, kueri klasifikasi simpul dapat sesederhana ini:

```
g.V("movie_1", "movie_2",
 "movie_3").properties("genre").with("Neptune#ml.classification")
```

Anda dapat mencampur properti vertex dan kelas dalam kueri menggunakan langkah `union()`:

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .union(
   properties("genre").with("Neptune#ml.classification"),
   properties("genre")
 )
```

Anda juga dapat membuat kueri tak terbatas seperti ini:

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V()
 .properties("genre").with("Neptune#ml.classification")
```

Anda dapat mengambil kelas simpul bersama-sama dengan vertex menggunakan langkah `select()` bersama dengan langkah `as()`:

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" ).as("vertex")
 .properties("genre").with("Neptune#ml.classification").as("properties")
 .select("vertex","properties")
```

Anda juga dapat menyaring pada kelas simpul, seperti yang digambarkan dalam contoh-contoh ini:

```
g.with("Neptune#ml.endpoint", "node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre").with("Neptune#ml.classification")
 .has(value, "Horror")

g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
```

```
  .properties("genre").with("Neptune#ml.classification")
  .has(value, P.eq("Action"))

g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V( "movie_1", "movie_2", "movie_3" )
  .properties("genre").with("Neptune#ml.classification")
  .has(value, P.within("Action", "Horror"))
```

Anda bisa mendapatkan skor kepercayaan klasifikasi node menggunakan `Neptune#ml.score` predikat:

```
 g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre", "Neptune#ml.score").with("Neptune#ml.classification")
```

Responsnya akan terlihat seperti ini:

```
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.01234567]
==>vp[genre->Crime]
==>vp[Neptune#ml.score->0.543210]
==>vp[genre->Comedy]
==>vp[Neptune#ml.score->0.10101]
```

Menggunakan inferensi induktif dalam kueri klasifikasi node

Misalkan Anda menambahkan simpul baru ke grafik yang ada, di buku catatan Jupyter, seperti ini:

```
%%gremlin
g.addV('label1').property(id,'101').as('newV')
 .V('1').as('oldV1')
 .V('2').as('oldV2')
 .addE('eLabel1').from('newV').to('oldV1')
 .addE('eLabel2').from('oldV2').to('newV')
```

Anda kemudian dapat menggunakan kueri inferensi induktif untuk mendapatkan genre dan skor kepercayaan yang mencerminkan node baru:

```
%%gremlin
```

```
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("genre", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
```

Namun, jika Anda menjalankan kueri beberapa kali, Anda mungkin mendapatkan hasil yang agak berbeda:

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.21365921]
```

Anda dapat membuat kueri deterministik yang sama:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("genre", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Dalam hal ini, hasilnya kira-kira sama setiap saat:

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
```

Kueri regresi simpul Gremlin di Neptunus ML

Regresi node mirip dengan klasifikasi node, kecuali bahwa nilai yang disimpulkan dari model regresi untuk setiap node adalah numerik. Anda dapat menggunakan kueri Gremlin yang sama untuk regresi node seperti untuk klasifikasi node kecuali untuk perbedaan berikut:

- Sekali lagi, di Neptune ML, simpul mengacu pada vertex.
- Langkah `properties()` mengambil bentuk, `properties().with("Neptune#ml.regression")` alih-alih `properties().with("Neptune#ml.classification")`.
- Predikat "`Neptune#ml.limit`" dan "`Neptune#ml.threshold`" tidak berlaku.
- Ketika Anda menyaring nilai, Anda harus menentukan nilai numerik.

Berikut ini adalah contoh kueri klasifikasi vertex:

```
g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1","movie_2","movie_3")
 .properties("revenue").with("Neptune#ml.regression")
```

Anda dapat menyaring nilai yang disimpulkan menggunakan model regresi, seperti yang digambarkan dalam contoh berikut:

```
g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1","movie_2","movie_3")
 .properties("revenue").with("Neptune#ml.regression")
 .value().is(P.gte(1600000))

g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1","movie_2","movie_3")
 .properties("revenue").with("Neptune#ml.regression")
 .hasValue(P.lte(1600000D))
```

Menggunakan inferensi induktif dalam kueri regresi simpul

Misalkan Anda menambahkan simpul baru ke grafik yang ada, di buku catatan Jupyter, seperti ini:

```
%%gremlin
g.addV('label1').property(id,'101').as('newV')
 .V('1').as('oldV1')
 .V('2').as('oldV2')
 .addE('eLabel1').from('newV').to('oldV1')
 .addE('eLabel2').from('oldV2').to('newV')
```

Anda kemudian dapat menggunakan kueri inferensi induktif untuk mendapatkan peringkat yang memperhitungkan node baru:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nr-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
```

Karena kueri tidak deterministik, kueri mungkin mengembalikan hasil yang agak berbeda jika Anda menjalankannya beberapa kali, berdasarkan lingkungan:

```
# First time
==>vp[rating->9.1]

# Second time
==>vp[rating->8.9]
```

Jika Anda membutuhkan hasil yang lebih konsisten, Anda dapat membuat kueri deterministik:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Sekarang hasilnya kira-kira sama setiap saat:

```
# First time
==>vp[rating->9.1]

# Second time
==>vp[rating->9.1]
```

Kueri klasifikasi tepi Gremlin di Neptunus ML

Untuk klasifikasi tepi Gremlin di Neptunus ML:

- Model ini dilatih pada satu properti tepi. Himpunan nilai unik dari properti ini disebut sebagai satu set kelas.
- Nilai properti kelas atau kategoris tepi dapat disimpulkan dari model klasifikasi tepi, yang berguna ketika properti ini belum melekat pada tepi.
- Untuk mengambil satu atau lebih kelas dari model klasifikasi tepi, Anda perlu menggunakan `with()` langkah dengan predikat, `"Neptune#ml.classification"` untuk mengkonfigurasi langkah. `properties()` Format output mirip dengan apa yang Anda harapkan jika itu adalah properti tepi.

> **Note**
>
> Klasifikasi tepi hanya berfungsi dengan nilai properti string. Itu berarti bahwa nilai properti numerik seperti `0` atau tidak `1` didukung, meskipun string setara `"0"` dan adalah. `"1"` Demikian pula, nilai properti Boolean `true` dan `false` tidak berfungsi, tetapi `"true"` dan `"false"` lakukan.

Berikut adalah contoh kueri klasifikasi tepi yang meminta skor kepercayaan menggunakan `Neptune#ml.score` predikat:

```
g.with("Neptune#ml.endpoint","edge-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by", "Neptune#ml.score").with("Neptune#ml.classification")
```

Responsnya akan terlihat seperti ini:

```
==>p[knows_by->"Family"]
==>p[Neptune#ml.score->0.01234567]
==>p[knows_by->"Friends"]
==>p[Neptune#ml.score->0.543210]
==>p[knows_by->"Colleagues"]
==>p[Neptune#ml.score->0.10101]
```

Sintaks kueri klasifikasi tepi Gremlin

Untuk grafik sederhana di `User` mana simpul kepala dan ekor, dan `Relationship` merupakan tepi yang menghubungkannya, contoh kueri klasifikasi tepi adalah:

```
g.with("Neptune#ml.endpoint","edge-classification-social-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by").with("Neptune#ml.classification")
```

Output dari kueri ini akan terlihat seperti berikut ini:

```
==>p[knows_by->"Family"]
==>p[knows_by->"Friends"]
==>p[knows_by->"Colleagues"]
```

Dalam query di atas, langkah `E()` dan `properties()` digunakan sebagai berikut:

- `E()`Langkah ini berisi kumpulan tepi yang ingin Anda ambil kelas dari model klasifikasi tepi:

  ```
  .E("relationship_1","relationship_2","relationship_3")
  ```

- `properties()`Langkah berisi kunci di mana model dilatih, dan `.with("Neptune#ml.classification")` harus menunjukkan bahwa ini adalah kueri inferensi HTML klasifikasi tepi.

Kunci properti ganda saat ini tidak didukung di langkah `properties().with("Neptune#ml.classification")`. Misalnya, kueri berikut menghasilkan pengecualian yang dilemparkan:

```
g.with("Neptune#ml.endpoint","edge-classification-social-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by", "other_label").with("Neptune#ml.classification")
```

Untuk pesan kesalahan tertentu, lihatDaftar pengecualian untuk kueri inferensi Gremlin Neptune ML.

Langkah `properties().with("Neptune#ml.classification")` dapat digunakan dalam kombinasi dengan semua langkah-langkah berikut:

- `value()`
- `value().is()`
- `hasValue()`

- has(value,"")
- key()
- key().is()
- hasKey()
- has(key,"")
- path()

Menggunakan inferensi induktif dalam kueri klasifikasi tepi

Misalkan Anda menambahkan tepi baru ke grafik yang ada, di buku catatan Jupyter, seperti ini:

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

Anda kemudian dapat menggunakan kueri inferensi induktif untuk mendapatkan skala yang memperhitungkan tepi baru:

```
%%gremlin
g.with("Neptune#ml.endpoint", "ec-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("scale", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
```

Karena kueri tidak deterministik, hasilnya akan agak berbeda jika Anda menjalankannya beberapa kali, berdasarkan lingkungan acak:

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.21365921]
```

Jika Anda membutuhkan hasil yang lebih konsisten, Anda dapat membuat kueri deterministik:

```
%%gremlin
g.with("Neptune#ml.endpoint", "ec-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("scale", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Sekarang hasilnya akan kurang lebih sama setiap kali Anda menjalankan kueri:

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]
```

Kueri regresi tepi Gremlin di Neptunus ML

Regresi tepi mirip dengan klasifikasi tepi, kecuali bahwa nilai yang disimpulkan dari model ML adalah numerik. Untuk regresi tepi, Neptunus ML mendukung kueri yang sama seperti untuk klasifikasi.

Poin penting yang perlu diperhatikan adalah:

- Anda perlu menggunakan predikat ML `"Neptune#ml.regression"` untuk mengonfigurasi `properties()` langkah untuk kasus penggunaan ini.
- `"Neptune#ml.limit"` `"Neptune#ml.threshold"`Predikat dan tidak berlaku dalam kasus penggunaan ini.
- Untuk memfilter nilai, Anda perlu menentukan nilai sebagai numerik.

Sintaks kueri regresi tepi Gremlin

Untuk grafik sederhana di mana `User` simpul kepala, `Movie` adalah simpul ekor, dan `Rated` tepi yang menghubungkannya, berikut adalah contoh kueri regresi tepi yang menemukan nilai peringkat numerik, disebut sebagai skor di sini, untuk tepi: `Rated`

```
g.with("Neptune#ml.endpoint","edge-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
```

```
  .E("rating_1","rating_2","rating_3")
  .properties("score").with("Neptune#ml.regression")
```

Anda juga dapat memfilter nilai yang disimpulkan dari model regresi ML. Untuk `Rated` tepi yang ada (dari `User` ke`Movie`) yang diidentifikasi oleh `"rating_1""rating_2","rating_3"`, dan, di mana properti tepi tidak `Score` ada untuk peringkat ini, Anda dapat menggunakan kueri seperti berikut `Score` untuk menyimpulkan tepi yang lebih besar dari atau sama dengan 9:

```
g.with("Neptune#ml.endpoint","edge-regression-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .E("rating_1","rating_2","rating_3")
  .properties("score").with("Neptune#ml.regression")
  .value().is(P.gte(9))
```

Menggunakan inferensi induktif dalam kueri regresi tepi

Misalkan Anda menambahkan tepi baru ke grafik yang ada, di buku catatan Jupyter, seperti ini:

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

Anda kemudian dapat menggunakan kueri inferensi induktif untuk mendapatkan skor yang memperhitungkan keunggulan baru:

```
%%gremlin
g.with("Neptune#ml.endpoint", "er-ep")
  .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
  .E('e101').properties("score")
  .with("Neptune#ml.regression")
  .with("Neptune#ml.inductiveInference")
```

Karena kueri tidak deterministik, hasilnya akan agak berbeda jika Anda menjalankannya beberapa kali, berdasarkan lingkungan acak:

```
# First time
==>ep[score->96]

# Second time
```

```
==>ep[score->91]
```

Jika Anda membutuhkan hasil yang lebih konsisten, Anda dapat membuat kueri deterministik:

```
%%gremlin
g.with("Neptune#ml.endpoint", "er-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("score")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Sekarang hasilnya akan kurang lebih sama setiap kali Anda menjalankan kueri:

```
# First time
==>ep[score->96]

# Second time
==>ep[score->96]
```

Kueri prediksi tautan Gremlin menggunakan model prediksi tautan di Neptunus ML

Model prediksi link dapat memecahkan masalah seperti berikut:

- Prediksi head-node: Mengingat simpul dan tipe tepi, dari simpul apa yang mungkin ditautkan oleh simpul itu?
- Prediksi ekor-simpul: Diberikan simpul dan label tepi, simpul apa yang mungkin ditautkan oleh simpul itu?

> **Note**
>
> Prediksi edge belum didukung di Neptune ML.

Untuk contoh di bawah ini, pertimbangkan grafik sederhana dengan simpul `User` dan `Movie` yang dihubungkan oleh tepi. `Rated`

Berikut adalah contoh kueri prediksi head-node, yang digunakan untuk memprediksi lima pengguna teratas yang paling mungkin menilai film,,`"movie_1"`, `"movie_2"` dan: `"movie_3"`

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .with("Neptune#ml.limit", 5)
 .V("movie_1", "movie_2", "movie_3")
 .in("rated").with("Neptune#ml.prediction").hasLabel("user")
```

Berikut ini adalah yang serupa untuk prediksi simpul ekor, yang digunakan untuk memprediksi lima film teratas yang kemungkinan akan `"user_1"` dinilai pengguna:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out("rated").with("Neptune#ml.prediction").hasLabel("movie")
```

Baik label tepi dan label simpul yang diprediksi diperlukan. Jika salah satu dihilangkan, maka akan melempar. Sebagai contoh, kueri berikut tanpa label vertex yang diprediksi melempar pengecualian:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out("rated").with("Neptune#ml.prediction")
```

Demikian pula, kueri berikut tanpa label edge melempar pengecualian:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out().with("Neptune#ml.prediction").hasLabel("movie")
```

Untuk pesan kesalahan spesifik yang ditampilkan pengecualian ini, lihat daftar pengecualian Neptunus ML.

Kueri prediksi link lainnya

Anda dapat menggunakan langkah `select()` dengan langkah `as()` untuk mengirim vertex yang diprediksi bersama-sama dengan vertex input:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1").as("source")
```

```
  .in("rated").with("Neptune#ml.prediction").hasLabel("user").as("target")
  .select("source","target")

g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V("user_1").as("source")
  .out("rated").with("Neptune#ml.prediction").hasLabel("movie").as("target")
  .select("source","target")
```

Anda dapat membuat kueri tak terbatas, seperti berikut ini:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V("user_1")
  .out("rated").with("Neptune#ml.prediction").hasLabel("movie")

g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V("movie_1")
  .in("rated").with("Neptune#ml.prediction").hasLabel("user")
```

Menggunakan inferensi induktif dalam kueri prediksi tautan

Misalkan Anda menambahkan simpul baru ke grafik yang ada, di buku catatan Jupyter, seperti ini:

```
%%gremlin
g.addV('label1').property(id,'101').as('newV1')
  .addV('label2').property(id,'102').as('newV2')
  .V('1').as('oldV1')
  .V('2').as('oldV2')
  .addE('eLabel1').from('newV1').to('oldV1')
  .addE('eLabel2').from('oldV2').to('newV2')
```

Anda kemudian dapat menggunakan kueri inferensi induktif untuk memprediksi node kepala, dengan mempertimbangkan node baru:

```
%%gremlin
g.with("Neptune#ml.endpoint", "lp-ep")
  .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
  .V('101').out("eLabel1")
  .with("Neptune#ml.prediction")
  .with("Neptune#ml.inductiveInference")
```

```
  .hasLabel("label2")
```

Hasil:

```
==>V[2]
```

Demikian pula, Anda dapat menggunakan kueri inferensi induktif untuk memprediksi simpul ekor, dengan mempertimbangkan simpul baru:

```
%%gremlin
g.with("Neptune#ml.endpoint", "lp-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('102').in("eLabel2")
 .with("Neptune#ml.prediction")
 .with("Neptune#ml.inductiveInference")
 .hasLabel("label1")
```

Hasil:

```
==>V[1]
```

Daftar pengecualian untuk kueri inferensi Gremlin Neptune ML

Ini adalah daftar lengkap pengecualian yang dapat terjadi saat menjalankan kueri inferensi Neptunus ML Gremlin. Pengecualian ini mencakup berbagai masalah, mulai dari masalah dengan peran IAM atau titik akhir yang ditentukan, hingga langkah Gremlin yang tidak didukung dan batasan jumlah kueri inferensi ML per kueri. Setiap entri menyertakan pesan terperinci yang menjelaskan masalah ini.

- **`BadRequestException`** – Kredensial untuk peran yang disediakan tidak dapat dimuat.

  Pesan: `Unable to load credentials for role: the specified IAM Role ARN.`
- **`BadRequestException`**— Peran IAM yang ditentukan tidak diizinkan untuk memanggil titik akhir SageMaker AI.

  Pesan: `User: the specified IAM Role ARN is not authorized to perform: sagemaker:InvokeEndpoint on resource: the specified endpoint.`
- **`BadRequestException`** – Titik akhir yang ditentukan tidak ada.

  Pesan: `Endpoint the specified endpoint not found.`

- **InternalFailureException**- Tidak dapat mengambil metadata inferensi induktif waktu-nyata Neptunus ML dari Amazon S3.

  Pesan: `Unable to fetch Neptune ML - Real-Time Inductive Inference metadata from S3. Check the permissions of the S3 bucket or if the Neptune instance can connect to S3.`

- **InternalFailureException**— Neptunus ML tidak dapat menemukan file metadata untuk inferensi induktif real-time di Amazon S3.

  Pesan: `Neptune ML cannot find the metadata file for Real-Time Inductive Inference in S3.`

- **InvalidParameterException** – Titik akhir yang ditentukan tidak valid secara sintaksis.

  Pesan: `Invalid endpoint provided for external service query.`

- **InvalidParameterException**— SageMaker Eksekusi IAM Role ARN yang ditentukan tidak valid secara sintaksis.

  Pesan: `Invalid IAM role ARN provided for external service query.`

- **InvalidParameterException** – Beberapa kunci properti ditentukan di langkah `properties()` dalam kueri.

  Pesan: `ML inference queries are currently supported for one property key.`

- **InvalidParameterException**— Beberapa label tepi ditentukan dalam kueri.

  Pesan: `ML inference are currently supported only with one edge label.`

- **InvalidParameterException**— Beberapa kendala label vertex ditentukan dalam kueri.

  Pesan: `ML inference are currently supported only with one vertex label constraint.`

- **InvalidParameterException** – Predikat `Neptune#ml.classification` dan `Neptune#ml.regression` hadir dalam kueri yang sama.

  Pesan: `Both regression and classification ML predicates cannot be specified in the query.`

- **InvalidParameterException** – Lebih dari satu label edge ditentukan di langkah `in()` atau `out()`dalam kueri prediksi link.

  Pesan: `ML inference are currently supported only with one edge label.`

- **InvalidParameterException**— Lebih dari satu kunci properti ditentukan dengan Neptunus #ml .score.

  Pesan: `Neptune ML inference queries are currently supported for one property key and one Neptune#ml.score property key.`

- **MissingParameterException** – Titik akhir tidak ditentukan dalam kueri atau sebagai parameter klaster DB.

  Pesan: `No endpoint provided for external service query.`

- MissingParameterException— Peran IAM eksekusi SageMaker AI tidak ditentukan dalam kueri atau sebagai parameter cluster DB.

  Pesan: `No IAM role ARN provided for external service query.`

- **MissingParameterException** – Kunci properti hilang dari langkah `properties()` dalam kueri.

  Pesan: `Property key needs to be specified using properties() step for ML inference queries.`

- **MissingParameterException** – Tidak ada label edge yang ditentukan dalam langkah `in()` atau `out()`dari kueri prediksi link.

  Pesan: `Edge label needs to be specified while using in() or out() step for ML inference queries.`

- **MissingParameterException**— Tidak ada kunci properti yang ditentukan dengan Neptunus #ml .score.

  Pesan: `Property key needs to be specified along with Neptune#ml.score property key while using the properties() step for Neptune ML inference queries.`

- **UnsupportedOperationException** – Langkah `both()` digunakan dalam kueri prediksi link.

  Pesan: `ML inference queries are currently not supported with both() step.`

- **UnsupportedOperationException** – Tidak ada label vertex yang diprediksikan ditentukan dalam langkah `has()` dengan langkah `in()` atau `out()` dalam kueri prediksi link.

  Pesan: `Predicted vertex label needs to be specified using has() step for ML inference queries.`

- **`UnsupportedOperationException`**— Kueri inferensi induktif Gremlin ML saat ini tidak didukung dengan langkah-langkah yang tidak dioptimalkan.

  Pesan: `Neptune ML - Real-Time Inductive Inference queries are currently not supported with Gremlin steps which are not optimized for Neptune. Check the Neptune User Guide for a list of Neptune-optimized steps.`
- **`UnsupportedOperationException`**— Kueri inferensi Neptunus ML saat ini tidak didukung dalam satu langkah. `repeat`

  Pesan: `Neptune ML inference queries are currently not supported inside a repeat step.`
- **`UnsupportedOperationException`**— Tidak lebih dari satu kueri inferensi Neptunus Neptunus saat ini didukung per kueri Gremlin.

  Pesan: `Neptune ML inference queries are currently supported only with one ML inference query per gremlin query.`

Kueri inferensi SPARQL di Neptunus ML

Neptunus ML memetakan grafik RDF ke dalam grafik properti untuk memodelkan Tugas ML. Saat ini, ia mendukung kasus penggunaan berikut:

- Klasifikasi objek — Memprediksi fitur kategoris suatu objek.
- Regresi objek — Memprediksi properti numerik dari suatu objek.
- Prediksi objek — Memprediksi objek yang diberikan subjek dan hubungan.
- Prediksi subjek — Memprediksi subjek yang diberi objek dan hubungan.

> **Note**
>
> Neptunus ML tidak mendukung klasifikasi subjek dan kasus penggunaan regresi dengan SPARQL.

Predikat Neptunus Neptunus yang digunakan dalam kueri inferensi SPARQL

Predikat berikut digunakan dengan inferensi SPARQL:

`neptune-ml:timeout`predikat

Menentukan batas waktu untuk koneksi dengan server jarak jauh. Jangan bingung dengan batas waktu permintaan kueri, yang merupakan jumlah waktu maksimum yang dapat diambil server untuk memenuhi permintaan.

Perhatikan bahwa jika batas waktu kueri terjadi sebelum batas waktu layanan yang ditentukan oleh `neptune-ml:timeout` predikat terjadi, sambungan layanan juga dibatalkan.

`neptune-ml:outputClass`predikat

`neptune-ml:outputClass`Predikat hanya digunakan untuk menentukan kelas objek yang diprediksi untuk prediksi objek atau subjek yang diprediksi untuk prediksi subect.

`neptune-ml:outputScore`predikat

`neptune-ml:outputScore`Predikat adalah angka positif yang mewakili kemungkinan bahwa output dari model pembelajaran mesin benar.

`neptune-ml:modelType`predikat

`neptune-ml:modelType`Predikat menentukan jenis model pembelajaran mesin yang dilatih:

- `OBJECT_CLASSIFICATION`
- `OBJECT_REGRESSION`
- `OBJECT_PREDICTION`
- `SUBJECT_PREDICTION`

`neptune-ml:input`predikat

`neptune-ml:input`Predikat mengacu pada daftar yang URIs digunakan sebagai input untuk Neptunus ML.

`neptune-ml:output`predikat

`neptune-ml:output`Predikat mengacu pada daftar set pengikatan di mana Neptunus ML mengembalikan hasil.

`neptune-ml:predicate`predikat

`neptune-ml:predicate`Predikat digunakan secara berbeda tergantung pada tugas yang dilakukan:

- Untuk prediksi objek atau subjek: mendefinisikan jenis predikat (tipe tepi atau hubungan).
- Untuk klasifikasi dan regresi objek: mendefinisikan literal (properti) yang ingin kita prediksi.

`neptune-ml:batchSize`predikat

`neptune-ml:batchSize`Menentukan ukuran input untuk panggilan layanan jarak jauh.

Contoh klasifikasi objek SPARQL

Untuk klasifikasi objek SPARQL di Neptunus ML, model dilatih pada salah satu nilai predikat. Ini berguna di mana predikat itu belum ada dengan subjek tertentu.

Hanya nilai predikat kategoris yang dapat disimpulkan menggunakan model klasifikasi objek.

Kueri berikut berusaha untuk < http://www.example.org/team > memprediksi nilai predikat untuk semua input tipe: `foaf:Person`

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_CLASSIFICATION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/team> ;
                      neptune-ml:output ?output .
  }
}
```

Kueri ini dapat disesuaikan sebagai berikut:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-account-balance-endpoint' ;
                      neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                      neptune-ml:batchSize "40"^^xsd:integer ;
                      neptune-ml:timeout "1000"^^xsd:integer ;

                      neptune-ml:modelType 'OBJECT_CLASSIFICATION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/team> ;
                      neptune-ml:output ?output .
  }
}
```

Contoh regresi objek SPARQL

Regresi objek mirip dengan klasifikasi objek, kecuali bahwa nilai predikat numerik disimpulkan dari model regresi untuk setiap node. Anda dapat menggunakan kueri SPARQL yang sama untuk regresi objek seperti untuk klasifikasi objek dengan pengecualian bahwa `the Neptune#ml.limit` dan predikat `Neptune#ml.threshold` tidak berlaku.

Kueri berikut berusaha untuk < http://www.example.org/accountbalance > memprediksi nilai predikat untuk semua input tipe: `foaf:Person`

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_REGRESSION' ;
                      neptune-ml:input ?input ;
```

```
                    neptune-ml:predicate <http://www.example.org/accountbalance> ;
                    neptune-ml:output ?output .
  }
}
```

Kueri ini dapat disesuaikan sebagai berikut:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-account-balance-endpoint' ;
                    neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                    neptune-ml:batchSize "40"^^xsd:integer ;
                    neptune-ml:timeout "1000"^^xsd:integer ;

                    neptune-ml:modelType 'OBJECT_REGRESSION' ;
                    neptune-ml:input ?input ;
                    neptune-ml:predicate <http://www.example.org/accountbalance> ;
                    neptune-ml:output ?output .
  }
}
```

Contoh prediksi objek SPARQL

Prediksi objek memprediksi nilai objek untuk subjek dan predikat tertentu.

Kueri prediksi objek berikut berupaya memprediksi film apa yang diinginkan oleh masukan tipe`foaf:Person`:

```
?x a foaf:Person .
?x   <http://www.example.org/likes> ?m .
?m a <http://www.example.org/movie> .

## Query
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_PREDICTION' ;
                    neptune-ml:input ?input ;
                    neptune-ml:predicate <http://www.example.org/likes> ;
                    neptune-ml:output ?output ;
                    neptune-ml:outputClass <http://www.example.org/movie> .
```

```
  }
}
```

Kueri itu sendiri dapat disesuaikan sebagai berikut:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-user-movie-prediction-
endpoint' ;
                      neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                      neptune-ml:limit "5"^^xsd:integer ;
                      neptune-ml:batchSize "40"^^xsd:integer ;
                      neptune-ml:threshold "0.1"^^xsd:double ;
                      neptune-ml:timeout "1000"^^xsd:integer ;
                      neptune-ml:outputScore ?score ;

                      neptune-ml:modelType 'OBJECT_PREDICTION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/likes> ;
                      neptune-ml:output ?output ;
                      neptune-ml:outputClass <http://www.example.org/movie> .
  }
}
```

Contoh prediksi subjek SPARQL

Prediksi subjek memprediksi subjek yang diberi predikat dan objek.

Misalnya, kueri berikut memprediksi siapa (tipe`foaf:User`) yang akan menonton film tertentu:

```
SELECT * WHERE { ?input (a foaf:Movie) .
    SERVICE neptune-ml:inference {
        neptune-ml:config neptune-ml:modelType 'SUBJECT_PREDICTION' ;
                          neptune-ml:input ?input ;
                          neptune-ml:predicate <http://aws.amazon.com/neptune/csv2rdf/
object_Property/rated> ;
                          neptune-ml:output ?output ;
                          neptune-ml:outputClass <http://aws.amazon.com/neptune/
csv2rdf/class/User> ;        }
}
```

Daftar pengecualian untuk kueri inferensi Neptunus ML SPARQL

- **BadRequestException**— Pesan: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects at least 1 value for the parameter *(parameter name)*, found zero.
- **BadRequestException**— Pesan: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects at most 1 value for the parameter *(parameter name)*, found *(a number)* values.
- **BadRequestException**— Pesan: Invalid predicate *(predicate name)* provided for external service http://aws.amazon.com/neptune/vocab/v01/services/ml#inference query.
- **BadRequestException**— Pesan:The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the predicate *(predicate name)* to be defined.
- **BadRequestException**— Pesan: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the value of (parameter) *(parameter name)* to be a variable, found: *(type)*"
- **BadRequestException**— Pesan:The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the input *(parameter name)* to be a constant, found: *(type)*.
- **BadRequestException**— Pesan:The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference is expected to return only 1 value.
- **BadRequestException**— Pesan:"The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference only allows StatementPatternNodes.
- **BadRequestException**— Pesan:The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference does not allow the predicate *(predicate name)*.
- **BadRequestException**— Pesan:The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference predicates cannot be variables, found: *(type)*.
- **BadRequestException**— Pesan:The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference predicates are expected to be part of the namespace *(namespace name)*, found: *(namespace name)*.

Referensi API manajemen Neptune ML

Daftar Isi

- Pemrosesan data menggunakan perintah dataprocessing
  - Membuat tugas pemrosesan data menggunakan perintah dataprocessing Neptune ML
  - Mendapatkan status job pemrosesan data menggunakan perintah dataprocessing Neptune ML
  - Menghentikan tugas pemrosesan data menggunakan perintah dataprocessing Neptune ML
  - Membuat daftar tugas pemrosesan data aktif menggunakan perintah dataprocessingNeptune ML
- Pelatihan model menggunakan perintah modeltraining
  - Membuat tugas pelatihan model menggunakan perintah modeltraining Neptune ML
  - Mendapatkan status tugas pelatihan model menggunakan perintah modeltraining Neptune ML
  - Menghentikan tugas pelatihan model menggunakan perintah modeltraining Neptune ML
  - Mendaftar tugas pelatihan model aktif menggunakan perintah modeltraining Neptune ML
- Transformasi model menggunakan modeltransform perintah
  - Membuat pekerjaan transformasi model menggunakan perintah Neptunus ML modeltransform
  - Mendapatkan status pekerjaan transformasi model menggunakan perintah Neptunus ML modeltransform
  - Menghentikan pekerjaan transformasi model menggunakan perintah Neptunus ML modeltransform
  - Daftar pekerjaan transformasi model aktif menggunakan perintah Neptunus ML modeltransform
- Mengelola titik akhir inferensi menggunakan perintah endpoints
  - Pembuatan titik akhir inferensi menggunakan perintah endpoints Neptune ML
  - Mendapatkan status titik akhir inferensi menggunakan perintah endpoints Neptune ML
  - Menghapus titik akhir instans menggunakan perintah endpoints Neptune ML
  - Membuat daftar titik akhir inferensi menggunakan perintah endpoints Neptune ML
- Kesalahan dan pengecualian API manajemen Neptunus ML

Pemrosesan data menggunakan perintah **`dataprocessing`**

Anda menggunakan perintah `dataprocessing` Neptune ML untuk membuat tugas pemrosesan data, memeriksa statusnya, menghentikannya, atau membuat daftar semua tugas pemrosesan data aktif.

Membuat tugas pemrosesan data menggunakan perintah **`dataprocessing`** Neptune ML

Perintah Neptunus `dataprocessing` ML yang khas untuk membuat pekerjaan baru terlihat seperti ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)"
      }'
```

Perintah untuk memulai pemrosesan ulang inkremental terlihat seperti ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for this job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)"
        "previousDataProcessingJobId" : "(the job ID of a previously completed job to
 update)"
}'
```

Parameter untuk pembuatan pekerjaan **`dataprocessing`**

- **`id`** – (Opsional) Pengidentifikasi unik untuk job baru.

Tipe: string. Default: UUID yang dihasilkan secara otomatis.

- **`previousDataProcessingJobId`**— (Opsional) ID pekerjaan dari pekerjaan pemrosesan data yang telah selesai dijalankan pada versi data sebelumnya.

  Tipe: string. Default: tidak ada.

  Catatan: Gunakan ini untuk pemrosesan data tambahan, untuk memperbarui model ketika data grafik telah berubah (tetapi tidak ketika data telah dihapus).

- **`inputDataS3Location`**— (Wajib) URI lokasi Amazon S3 tempat Anda ingin SageMaker AI mengunduh data yang diperlukan untuk menjalankan pekerjaan pemrosesan data.

  Tipe: string.

- **`processedDataS3Location`**— (Wajib) URI lokasi Amazon S3 tempat Anda ingin SageMaker AI menyimpan hasil pekerjaan pemrosesan data.

  Tipe: string.

- **`sagemakerIamRoleArn`**— (Opsional) ARN dari peran IAM untuk SageMaker eksekusi AI.

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.

- **`neptuneIamRoleArn`**— (Opsional) Nama Sumber Daya Amazon (ARN) dari peran IAM yang dapat diasumsikan SageMaker AI untuk melakukan tugas atas nama Anda.

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.

- **`processingInstanceType`**— (Opsional) Jenis instans ML yang digunakan selama pemrosesan data. Memorinya harus cukup besar untuk menahan set data yang diproses.

  Tipe: string. Default: `ml.r5` tipe terkecil yang memorinya sepuluh kali lebih besar dari ukuran data grafik yang diekspor pada disk.

  Catatan: Neptunus ML dapat memilih jenis instans secara otomatis. Lihat Memilih instance untuk pemrosesan data.

- **`processingInstanceVolumeSizeInGB`** – (Opsional) Ukuran volume disk dari instans pemrosesan. Data input dan data yang diproses disimpan pada disk, sehingga ukuran volume harus cukup besar untuk menahan kedua set data.

Tipe: integer. Default: `0`.

Catatan: Jika tidak ditentukan atau 0, Neptunus ML memilih ukuran volume secara otomatis berdasarkan ukuran data.

- **`processingTimeOutInSeconds`** – (Opsional) Timeout dalam hitungan detik untuk tugas pemrosesan data.

  Tipe: integer. Default: `86,400` (1 hari).

- **`modelType`**— (Opsional) Salah satu dari dua jenis model yang saat ini didukung Neptunus ML: model grafik heterogen `heterogeneous` (), dan grafik pengetahuan (). `kge`

  Tipe: string. Default: tidak ada.

  Catatan: Jika tidak ditentukan, Neptunus ML memilih jenis model secara otomatis berdasarkan data.

- **`configFileName`** – (Opsional) File spesifikasi data yang menjelaskan cara memuat data grafik yang diekspor untuk pelatihan. File secara otomatis dihasilkan oleh kit alat ekspor Neptune.

  Tipe: string. Default: `training-data-configuration.json`.

- **`subnets`**— (Opsional) Subnet di VPC Neptunus. IDs

  Tipe: daftar string. Default: tidak ada.

- **`securityGroupIds`**— (Opsional) Grup keamanan VPC. IDs

  Tipe: daftar string. Default: tidak ada.

- **`volumeEncryptionKMSKey`**— (Opsional) Kunci AWS Key Management Service (AWS KMS) yang digunakan SageMaker AI untuk mengenkripsi data pada volume penyimpanan yang dilampirkan ke instance komputasi ML yang menjalankan tugas pemrosesan.

  Tipe: string. Default: tidak ada.

- **`enableInterContainerTrafficEncryption`**— (Opsional) Aktifkan atau nonaktifkan enkripsi lalu lintas antar-kontainer dalam pelatihan atau pekerjaan penyetelan hiper-parameter.

  Jenis: boolean. Default: Benar.

> **Note**
>
> `enableInterContainerTrafficEncryption`Parameter ini hanya tersedia dalam rilis mesin 1.2.0.2.R3.

- **`s3OutputEncryptionKMSKey`**— (Opsional) Kunci AWS Key Management Service (AWS KMS) yang digunakan SageMaker AI untuk mengenkripsi output dari pekerjaan pelatihan.

  Tipe: string. Default: tidak ada.

Mendapatkan status job pemrosesan data menggunakan perintah **`dataprocessing`** Neptune ML

Perintah `dataprocessing` Neptune ML sampel untuk status tugas terlihat seperti ini:

```
curl -s \
  "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)" \
  | python -m json.tool
```

Parameter untuk status tugas **`dataprocessing`**

- **`id`** – (Wajib) Pengenal unik tugas pemrosesan data.

  Tipe: string.
- **`neptuneIamRoleArn`**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.

Menghentikan tugas pemrosesan data menggunakan perintah **`dataprocessing`** Neptune ML

Perintah `dataprocessing` Neptune ML sampel untuk menghentikan tugas terlihat seperti ini:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)"
```

Atau ini:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)?clean=true"
```

Parameter untuk tugas berhenti **dataprocessing**

- **id** – (Wajib) Pengenal unik tugas pemrosesan data.

  Tipe: string.
- **neptuneIamRoleArn**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.
- **clean** – (Opsional) Bendera ini menetapkan bahwa semua artefak Amazon S3 harus dihapus ketika tugas dihentikan.

  Tipe: Boolean. Default: `FALSE`.

Membuat daftar tugas pemrosesan data aktif menggunakan perintah **dataprocessing**Neptune ML

Perintah `dataprocessing` Neptune ML sampel untuk membuat daftar tugas aktif terlihat seperti ini:

```
curl -s "https://(your Neptune endpoint)/ml/dataprocessing"
```

Atau ini:

```
curl -s "https://(your Neptune endpoint)/ml/dataprocessing?maxItems=3"
```

Parameter untuk tugas daftar **dataprocessing**

- **maxItems** – (Opsional) Jumlah maksimum item yang akan dikembalikan.

  Tipe: integer. Default: `10`. Nilai maksimum yang diperbolehkan: `1024`.
- **neptuneIamRoleArn**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.

Pelatihan model menggunakan perintah **`modeltraining`**

Anda menggunakan perintah `modeltraining` Neptune ML untuk membuat tugas pelatihan model, memeriksa statusnya, menghentikannya, atau membuat daftar semua tugas pelatihan model aktif.

Membuat tugas pelatihan model menggunakan perintah **`modeltraining`** Neptune ML

Perintah Neptunus `modeltraining` Neptunus untuk membuat pekerjaan yang sama sekali baru terlihat seperti ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
      }'
```

Perintah Neptunus `modeltraining` Neptunus untuk membuat pekerjaan pembaruan untuk pelatihan model tambahan terlihat seperti ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "previousModelTrainingJobId" : "(the job ID of a completed model-training job
 to update)",
      }'
```

Perintah Neptunus `modeltraining` Neptunus untuk membuat pekerjaan baru dengan implementasi model kustom yang disediakan pengguna terlihat seperti:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
```

```
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Parameter untuk pembuatan pekerjaan **`modeltraining`**

- **`id`** – (Opsional) Pengidentifikasi unik untuk job baru.

  Tipe: string. Default: UUID yang dihasilkan secara otomatis.
- **`dataProcessingJobId`** – (Wajib) Tugas ID dari tugas pemrosesan data yang telah selesai yang telah membuat data yang akan dikerjakan oleh pelatihan.

  Tipe: string.
- **`trainModelS3Location`** – (Wajib) Lokasi di Amazon S3 di mana artefak model akan disimpan.

  Tipe: string.
- **`previousModelTrainingJobId`**— (Opsional) ID pekerjaan dari pekerjaan pelatihan model selesai yang ingin Anda perbarui secara bertahap berdasarkan data yang diperbarui.

  Tipe: string. Default: tidak ada.
- **`sagemakerIamRoleArn`**— (Opsional) ARN dari peran IAM untuk SageMaker eksekusi AI.

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.
- **`neptuneIamRoleArn`**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.

- **`modelName`** – (Opsional) Jenis model untuk pelatihan. Secara default model ML secara otomatis didasarkan pada `modelType` yang digunakan dalam pemrosesan data, tetapi Anda dapat menentukan jenis model yang berbeda di sini.

  Tipe: string. Default: `rgcn` untuk grafik heterogen dan `kge` untuk grafik pengetahuan. Nilai yang valid: Untuk grafik heterogen: `rgcn`. Untuk `kge` grafik:`transe`,`distmult`, atau`rotate`. Untuk implementasi model khusus:`custom`.

- **`baseProcessingInstanceType`**— (Opsional) Jenis instance ML yang digunakan dalam mempersiapkan dan mengelola pelatihan model ML.

  Tipe: string. Catatan: Ini adalah instance CPU yang dipilih berdasarkan persyaratan memori untuk memproses data dan model pelatihan. Lihat Memilih contoh untuk pelatihan model dan transformasi model.

- **`trainingInstanceType`** – (Opsional) Jenis instans ML yang digunakan untuk pelatihan model Semua model Neptune ML mendukung pelatihan CPU, GPU, dan MultiGPU.

  Tipe: string. Default: `ml.p3.2xlarge`.

  Catatan: Memilih jenis instans yang tepat untuk pelatihan tergantung pada jenis tugas, ukuran grafik, dan anggaran Anda. Lihat Memilih contoh untuk pelatihan model dan transformasi model.

- **`trainingInstanceVolumeSizeInGB`** – (Opsional) Ukuran volume disk dari instans pelatihan. Input data dan model output disimpan dalam disk, sehingga ukuran volume harus cukup besar untuk menahan kedua set data.

  Tipe: integer. Default: `0`.

  Catatan: Jika tidak ditentukan atau 0, Neptunus ML memilih ukuran volume disk berdasarkan rekomendasi yang dihasilkan dalam langkah pemrosesan data. Lihat Memilih contoh untuk pelatihan model dan transformasi model.

- **`trainingTimeOutInSeconds`** – (Opsional) Timeout dalam hitungan detik untuk tugas pelatihan.

  Tipe: integer. Default: `86,400` (1 hari).

- **`maxHPONumberOfTrainingJobs`** – Jumlah total maksimum dari tugas pelatihan untuk memulai tugas penyetelan hyperparameter.

Tipe: integer. Default: 2.

Catatan: Neptunus ML secara otomatis menyetel parameter hiper model pembelajaran mesin. Untuk mendapatkan model yang berkinerja baik, gunakan setidaknya 10 pekerjaan (dengan kata lain, atur `maxHPONumberOfTrainingJobs` ke 10). Secara umum, semakin banyak tuning berjalan, semakin baik hasilnya.

- **`maxHPOParallelTrainingJobs`** – Jumlah maksimum dari tugas pelatihan untuk memulai tugas penyetelan hyperparameter.

  Tipe: integer. Default: 2.

  Catatan: Jumlah pekerjaan paralel yang dapat Anda jalankan dibatasi oleh sumber daya yang tersedia pada instans pelatihan Anda.

- **`subnets`**— (Opsional) Subnet di VPC Neptunus. IDs

  Tipe: daftar string. Default: tidak ada.

- **`securityGroupIds`**— (Opsional) Grup keamanan VPC. IDs

  Tipe: daftar string. Default: tidak ada.

- **`volumeEncryptionKMSKey`**— (Opsional) Kunci AWS Key Management Service (AWS KMS) yang digunakan SageMaker AI untuk mengenkripsi data pada volume penyimpanan yang dilampirkan ke instance komputasi ML yang menjalankan tugas pelatihan.

  Tipe: string. Default: tidak ada.

- **`s3OutputEncryptionKMSKey`**— (Opsional) Kunci AWS Key Management Service (AWS KMS) yang digunakan SageMaker AI untuk mengenkripsi output dari pekerjaan pemrosesan.

  Tipe: string. Default: tidak ada.

- **`enableInterContainerTrafficEncryption`**— (Opsional) Aktifkan atau nonaktifkan enkripsi lalu lintas antar-kontainer dalam pelatihan atau pekerjaan penyetelan hiper-parameter.

  Jenis: boolean. Default: Benar.

> **Note**
>
> `enableInterContainerTrafficEncryption`Parameter ini hanya tersedia dalam rilis mesin 1.2.0.2.R3.

- **`enableManagedSpotTraining`**— (Opsional) Mengoptimalkan biaya pelatihan model machine learning dengan menggunakan instans spot Amazon Elastic Compute Cloud. Untuk informasi selengkapnya, lihat Pelatihan Spot Terkelola di Amazon SageMaker.

  Tipe: Boolean. Default: Salah.

- **`customModelTrainingParameters`**— (Opsional) Konfigurasi untuk pelatihan model kustom. Ini adalah objek JSON dengan bidang-bidang berikut:
  - **`sourceS3DirectoryPath`**— (Wajib) Jalur ke lokasi Amazon S3 tempat modul Python yang mengimplementasikan model Anda berada. Ini harus menunjuk ke lokasi Amazon S3 yang valid yang berisi, setidaknya, skrip pelatihan, skrip transformasi, dan file. `model-hpo-configuration.json`
  - **`trainingEntryPointScript`**— (Opsional) Nama titik masuk dalam modul skrip Anda yang melakukan pelatihan model dan mengambil hyperparameters sebagai argumen baris perintah, termasuk hyperparameters tetap.

    Default: `training.py`.
  - **`transformEntryPointScript`**— (Opsional) Nama titik masuk dalam modul skrip Anda yang harus dijalankan setelah model terbaik dari pencarian hyperparameter telah diidentifikasi, untuk menghitung artefak model yang diperlukan untuk penerapan model. Itu harus dapat berjalan tanpa argumen baris perintah.

    Default: `transform.py`.
- **`maxWaitTime`**— (Opsional) Waktu maksimum untuk menunggu, dalam hitungan detik, saat melakukan pelatihan model menggunakan instance spot. Harus lebih besar dari`trainingTimeOutInSeconds`.

  Tipe: integer.

Mendapatkan status tugas pelatihan model menggunakan perintah **`modeltraining`** Neptune ML

Perintah `modeltraining` Neptune ML sampel untuk status tugas terlihat seperti ini:

```
curl -s \
  "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)" \
  | python -m json.tool
```

Parameter untuk status tugas **`modeltraining`**

- **`id`** – (Wajib) Pengenal unik tugas pelatihan model.

  Tipe: string.

- **`neptuneIamRoleArn`**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.

Menghentikan tugas pelatihan model menggunakan perintah **`modeltraining`** Neptune ML

Perintah `modeltraining` Neptune ML sampel untuk menghentikan tugas terlihat seperti ini:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)"
```

Atau ini:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)?clean=true"
```

Parameter untuk tugas berhenti **`modeltraining`**

- **`id`** – (Wajib) Pengenal unik tugas pelatihan model.

  Tipe: string.

- **`neptuneIamRoleArn`**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.

- **`clean`** – (Opsional) Bendera ini menetapkan bahwa semua artefak Amazon S3 harus dihapus ketika tugas dihentikan.

  Tipe: Boolean. Default: `FALSE`.

Mendaftar tugas pelatihan model aktif menggunakan perintah **`modeltraining`** Neptune ML

Perintah `modeltraining` Neptune ML sampel untuk membuat daftar tugas aktif terlihat seperti ini:

```
curl -s "https://(your Neptune endpoint)/ml/modeltraining" | python -m json.tool
```

Atau ini:

```
curl -s "https://(your Neptune endpoint)/ml/modeltraining?maxItems=3" | python -m
 json.tool
```

Parameter untuk tugas daftar **`modeltraining`**

- **`maxItems`** – (Opsional) Jumlah maksimum item yang akan dikembalikan.

  Tipe: integer. Default: `10`. Nilai maksimum yang diperbolehkan: `1024`.
- **`neptuneIamRoleArn`**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.

Transformasi model menggunakan **`modeltransform`** perintah

Anda menggunakan perintah Neptunus `modeltransform` ML untuk membuat pekerjaan transformasi model, memeriksa statusnya, menghentikannya, atau mencantumkan semua pekerjaan transformasi model yang aktif.

Membuat pekerjaan transformasi model menggunakan perintah Neptunus ML **`modeltransform`**

Perintah Neptunus `modeltransform` Neptunus untuk membuat pekerjaan transformasi inkremental, tanpa pelatihan ulang model, terlihat seperti ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "dataProcessingJobId" : "(the job-id of a completed data-processing job)",
        "mlModelTrainingJobId" : "(the job-id of a completed model-training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform"
      }'
```

Perintah Neptunus `modeltransform` ML untuk membuat pekerjaan dari pekerjaan pelatihan AI yang SageMaker telah selesai terlihat seperti ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "trainingJobName" : "(name of a completed SageMaker AI training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform",
        "baseProcessingInstanceType" : ""
      }'
```

Perintah Neptunus `modeltransform` Neptunus untuk membuat pekerjaan yang menggunakan implementasi model kustom terlihat seperti:

```
curl \
```

```
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "trainingJobName" : "(name of a completed SageMaker AI training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
        "customModelTransformParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Parameter untuk pembuatan pekerjaan **`modeltransform`**

- **`id`** – (Opsional) Pengidentifikasi unik untuk job baru.

  Tipe: string. Default: UUID yang dihasilkan secara otomatis.

- **`dataProcessingJobId`**— Id pekerjaan dari pekerjaan pemrosesan data yang diselesaikan.

  Tipe: string.

  Catatan: Anda harus menyertakan keduanya `dataProcessingJobId` dan`mlModelTrainingJobId`, atau`trainingJobName`.

- **`mlModelTrainingJobId`**— Id pekerjaan dari pekerjaan pelatihan model yang diselesaikan.

  Tipe: string.

  Catatan: Anda harus menyertakan keduanya `dataProcessingJobId` dan`mlModelTrainingJobId`, atau`trainingJobName`.

- **`trainingJobName`**— Nama pekerjaan pelatihan SageMaker AI yang telah selesai.

  Tipe: string.

  Catatan: Anda harus menyertakan parameter `dataProcessingJobId` dan `mlModelTrainingJobId` parameter, atau `trainingJobName` parameternya.

- **`sagemakerIamRoleArn`**— (Opsional) ARN dari peran IAM untuk SageMaker eksekusi AI.

Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.

- **`neptuneIamRoleArn`**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.

- **`customModelTransformParameters`** — (Opsional) Informasi konfigurasi untuk transformasi model menggunakan model kustom. `customModelTransformParameters`Objek berisi bidang berikut, yang harus memiliki nilai yang kompatibel dengan parameter model yang disimpan dari pekerjaan pelatihan:

  - **`sourceS3DirectoryPath`**— (Wajib) Jalur ke lokasi Amazon S3 tempat modul Python yang mengimplementasikan model Anda berada. Ini harus menunjuk ke lokasi Amazon S3 yang valid yang berisi, setidaknya, skrip pelatihan, skrip transformasi, dan file. `model-hpo-configuration.json`
  - **`transformEntryPointScript`**— (Opsional) Nama titik masuk dalam modul skrip Anda yang harus dijalankan setelah model terbaik dari pencarian hyperparameter telah diidentifikasi, untuk menghitung artefak model yang diperlukan untuk penerapan model. Itu harus dapat berjalan tanpa argumen baris perintah.

    Default: `transform.py`.

- **`baseProcessingInstanceType`**— (Opsional) Jenis instance ML yang digunakan dalam mempersiapkan dan mengelola pelatihan model ML.

  Tipe: string. Catatan: Ini adalah instance CPU yang dipilih berdasarkan persyaratan memori untuk memproses data dan model transformasi. Lihat Memilih contoh untuk pelatihan model dan transformasi model.

- **`baseProcessingInstanceVolumeSizeInGB`** – (Opsional) Ukuran volume disk dari instans pelatihan. Input data dan model output disimpan dalam disk, sehingga ukuran volume harus cukup besar untuk menahan kedua set data.

  Tipe: integer. Default: `0`.

  Catatan: Jika tidak ditentukan atau 0, Neptunus ML memilih ukuran volume disk berdasarkan rekomendasi yang dihasilkan dalam langkah pemrosesan data. Lihat Memilih contoh untuk pelatihan model dan transformasi model.

- **`subnets`**— (Opsional) Subnet di VPC Neptunus. IDs

  Tipe: daftar string. Default: tidak ada.

- **`securityGroupIds`**— (Opsional) Grup keamanan VPC. IDs

  Tipe: daftar string. Default: tidak ada.

- **`volumeEncryptionKMSKey`**— (Opsional) Kunci AWS Key Management Service (AWS KMS) yang digunakan SageMaker AI untuk mengenkripsi data pada volume penyimpanan yang dilampirkan ke instance komputasi ML yang menjalankan tugas transformasi.

  Tipe: string. Default: tidak ada.

- **`enableInterContainerTrafficEncryption`**— (Opsional) Aktifkan atau nonaktifkan enkripsi lalu lintas antar kontainer dalam pelatihan atau pekerjaan penyetelan hiper-parameter.

  Jenis: boolean. Default: Benar.

  > **Note**
  >
  > `enableInterContainerTrafficEncryption`Parameter ini hanya tersedia dalam rilis mesin 1.2.0.2.R3.

- **`s3OutputEncryptionKMSKey`**— (Opsional) Kunci AWS Key Management Service (AWS KMS) yang digunakan SageMaker AI untuk mengenkripsi output dari pekerjaan pemrosesan.

  Tipe: string. Default: tidak ada.

Mendapatkan status pekerjaan transformasi model menggunakan perintah Neptunus ML **`modeltransform`**

Perintah `modeltransform` Neptune ML sampel untuk status tugas terlihat seperti ini:

```
curl -s \
  "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)" \
  | python -m json.tool
```

Parameter untuk status tugas **`modeltransform`**

- **`id`**— (Wajib) Pengidentifikasi unik dari pekerjaan transformasi model.

Tipe: string.

- **neptuneIamRoleArn**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.

Menghentikan pekerjaan transformasi model menggunakan perintah Neptunus ML **modeltransform**

Perintah `modeltransform` Neptune ML sampel untuk menghentikan tugas terlihat seperti ini:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)"
```

Atau ini:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)?clean=true"
```

Parameter untuk tugas berhenti **modeltransform**

- **id**— (Wajib) Pengidentifikasi unik dari pekerjaan transformasi model.

  Tipe: string.

- **neptuneIamRoleArn**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.

- **clean** – (Opsional) Bendera ini menetapkan bahwa semua artefak Amazon S3 harus dihapus ketika tugas dihentikan.

  Tipe: Boolean. Default: `FALSE`.

Daftar pekerjaan transformasi model aktif menggunakan perintah Neptunus ML **`modeltransform`**

Perintah `modeltransform` Neptune ML sampel untuk membuat daftar tugas aktif terlihat seperti ini:

```
curl -s "https://(your Neptune endpoint)/ml/modeltransform" | python -m json.tool
```

Atau ini:

```
curl -s "https://(your Neptune endpoint)/ml/modeltransform?maxItems=3" | python -m
 json.tool
```

Parameter untuk tugas daftar **`modeltransform`**

- **`maxItems`** – (Opsional) Jumlah maksimum item yang akan dikembalikan.

  Tipe: integer. Default: `10`. Nilai maksimum yang diperbolehkan: `1024`.
- **`neptuneIamRoleArn`**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan terjadi.

Mengelola titik akhir inferensi menggunakan perintah **endpoints**

Anda menggunakan perintah `endpoints` Neptune ML untuk membuat titik akhir inferensi, memeriksa statusnya, menghapus, atau membuat daftar titik akhir inferensi yang ada.

Pembuatan titik akhir inferensi menggunakan perintah **endpoints** Neptune ML

Perintah Neptunus `endpoints` Neptunus untuk membuat titik akhir inferensi dari model yang dibuat oleh pekerjaan pelatihan terlihat seperti ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Perintah Neptunus `endpoints` Neptunus untuk memperbarui titik akhir inferensi yang ada dari model yang dibuat oleh pekerjaan pelatihan terlihat seperti ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "update" : "true",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Perintah Neptunus `endpoints` Neptunus untuk membuat titik akhir inferensi dari model yang dibuat oleh pekerjaan transformasi model terlihat seperti ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTransformJobId": "(the model-training job-id of a completed job)"
      }'
```

Perintah Neptunus `endpoints` Neptunus untuk memperbarui titik akhir inferensi yang ada dari model yang dibuat oleh pekerjaan transformasi model terlihat seperti ini:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "update" : "true",
        "mlModelTransformJobId": "(the model-training job-id of a completed job)"
      }'
```

Parameter untuk pembuatan titik akhir inferensi **`endpoints`**

- **`id`** – (Opsional) Pengidentifikasi unik untuk titik akhir inferensi baru.

  Tipe: string. Default: Nama berstampel waktu yang otomatis dihasilkan.

- **`mlModelTrainingJobId`**— Id pekerjaan dari pekerjaan pelatihan model yang telah diselesaikan yang telah menciptakan model yang akan ditunjukkan oleh titik akhir inferensi.

  Tipe: string.

  Catatan: Anda harus menyediakan salah satu `mlModelTrainingJobId` atau`mlModelTransformJobId`.

- **`mlModelTransformJobId`**— Id pekerjaan dari pekerjaan transformasi model yang telah selesai.

  Tipe: string.

  Catatan: Anda harus menyediakan salah satu `mlModelTrainingJobId` atau`mlModelTransformJobId`.

- **`update`**— (Opsional) Jika ada, parameter ini menunjukkan bahwa ini adalah permintaan pembaruan.

  Tipe: Boolean. Default: `false`

  Catatan: Anda harus menyediakan salah satu `mlModelTrainingJobId` atau`mlModelTransformJobId`.

- **`neptuneIamRoleArn`**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan dilemparkan.

- **`modelName`** – (Opsional) Jenis model untuk latihan. Secara default model ML secara otomatis didasarkan pada `modelType` yang digunakan dalam pemrosesan data, tetapi Anda dapat menentukan jenis model yang berbeda di sini.

  Tipe: string. Default: `rgcn` untuk grafik heterogen dan `kge` untuk grafik pengetahuan. Nilai yang valid: Untuk grafik heterogen: `rgcn`. Untuk grafik pengetahuan: `kge`, `transe`, `distmult`, atau `rotate`.

- **`instanceType`** – (Opsional) Jenis instans ML yang digunakan untuk servis online.

  Tipe: string. Default: `ml.m5.xlarge`.

  Catatan: Memilih instans ML untuk titik akhir inferensi tergantung pada jenis tugas, ukuran grafik, dan anggaran Anda. Lihat Memilih instance untuk titik akhir inferensi.

- **`instanceCount`** – (Opsional) Jumlah minimum instans Amazon EC2 untuk mendeploy ke titik akhir untuk prediksi.

  Tipe: integer. Default: 1.

- **`volumeEncryptionKMSKey`**— (Opsional) Kunci AWS Key Management Service (AWS KMS) yang digunakan SageMaker AI untuk mengenkripsi data pada volume penyimpanan yang dilampirkan ke instance komputasi HTML yang menjalankan titik akhir.

  Tipe: string. Default: tidak ada.

Mendapatkan status titik akhir inferensi menggunakan perintah **`endpoints`** Neptune ML

Perintah `endpoints` Neptune ML sampel untuk status titik akhir instans terlihat seperti ini:

```
curl -s \
  "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)" \
  | python -m json.tool
```

Parameter untuk status titik akhir instans **`endpoints`**

- **`id`** – (Wajib) Pengenal unik dari titik akhir inferensi.

Tipe: string.

- **neptuneIamRoleArn**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan dilemparkan.

Menghapus titik akhir instans menggunakan perintah **endpoints** Neptune ML

Perintah `endpoints` Neptune ML sampel untuk menghapus titik akhir instans terlihat seperti ini:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)"
```

Atau ini:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)?
clean=true"
```

Parameter untuk **endpoints** menghapus sebuah titik akhir inferensi

- **id** – (Wajib) Pengenal unik dari titik akhir inferensi.

  Tipe: string.
- **neptuneIamRoleArn**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan dilemparkan.
- **clean**— (Opsional) Menunjukkan bahwa semua artefak yang terkait dengan titik akhir ini juga harus dihapus.

  Tipe: Boolean. Default: `FALSE`.

Membuat daftar titik akhir inferensi menggunakan perintah **endpoints** Neptune ML

Perintah Neptunus Neptunus `endpoints` untuk mencantumkan titik akhir inferensi terlihat seperti ini:

```
curl -s "https://(your Neptune endpoint)/ml/endpoints" \
  | python -m json.tool
```

Atau ini:

```
curl -s "https://(your Neptune endpoint)/ml/endpoints?maxItems=3" \
  | python -m json.tool
```

Parameter untuk **`dataprocessing`** membuat daftar titik akhir inferensi

- **`maxItems`** – (Opsional) Jumlah maksimum item yang akan dikembalikan.

  Tipe: integer. Default: `10`. Nilai maksimum yang diperbolehkan: `1024`.
- **`neptuneIamRoleArn`**— (Opsional) ARN dari peran IAM yang menyediakan akses Neptunus ke sumber daya AI dan Amazon S3. SageMaker

  Tipe: string. Catatan: Ini harus tercantum dalam grup parameter klaster DB Anda atau kesalahan akan dilemparkan.

Kesalahan dan pengecualian API manajemen Neptunus ML

Semua pengecualian API manajemen Neptunus ML mengembalikan kode HTTP 400. Setelah menerima salah satu pengecualian ini, perintah yang menghasilkan pengecualian tidak boleh dicoba lagi.

- **`MissingParameterException`**— Pesan kesalahan:

  ```
  Required credentials are missing. Please add IAM role to the cluster or pass as a parameter to this request.
  ```

- **`InvalidParameterException`**— Pesan kesalahan:
  - `Invalid ML instance type.`
  - `Invalid ID provided. ID can be 1-48 alphanumeric characters.`
  - `Invalid ID provided. Must contain only letters, digits, or hyphens.`
  - `Invalid ID provided. Please check whether a resource with the given ID exists.`
  - `Another resource with same ID already exists. Please use a new ID.`
  - `Failed to stop the job because it has already completed or failed.`
- **`BadRequestException`**— Pesan kesalahan:
  - `Invalid S3 URL or incorrect S3 permissions. Please check your S3 configuration.`
  - `Provided ModelTraining job has not completed.`
  - `Provided SageMaker AI Training job has not completed.`
  - `Provided MLDataProcessing job is not completed.`
  - `Provided MLModelTraining job doesn't exist.`
  - `Provided ModelTransformJob doesn't exist.`
  - `Unable to find SageMaker AI resource. Please check your input.`

Batas Neptune ML

- Jenis inferensi yang saat ini didukung adalah klasifikasi simpul, regresi simpul, klasifikasi tepi, regresi tepi, dan prediksi tautan (lihat). Kemampuan Neptune ML
- Ukuran grafik maksimum yang dapat didukung Neptunus ML tergantung pada jumlah memori dan penyimpanan yang diperlukan selama persiapan data, pelatihan model, dan inferensi.
  - Ukuran maksimum memori instance pemrosesan data SageMaker AI adalah 768 GB. Akibatnya, tahap pemrosesan data gagal jika membutuhkan memori lebih dari 768 GB.
  - Ukuran maksimum memori instans pelatihan SageMaker AI adalah 732 GB. Akibatnya, tahap pelatihan gagal jika membutuhkan memori lebih dari 732 GB.
- Ukuran maksimum payload inferensi untuk titik akhir SageMaker AI adalah 6 MiB. Akibatnya, inferensi induktif gagal jika muatan subgraf melebihi ukuran ini.
- Neptunus ML saat ini hanya tersedia di Wilayah di mana Neptunus dan layanan lain yang bergantung padanya (seperti, Amazon API AWS Lambda Gateway dan Amazon AI) semuanya didukung. SageMaker

  Ada perbedaan di Tiongkok (Beijing) dan Tiongkok (Ningxia) yang berkaitan dengan penggunaan default otentikasi IAM, seperti yang dijelaskan di sini bersama dengan perbedaan lainnya.
- Titik akhir inferensi prediksi tautan yang diluncurkan oleh Neptunus ML saat ini hanya dapat memprediksi kemungkinan tautan dengan node yang ada dalam grafik selama pelatihan.

  Misalnya, perhatikan grafik dengan `User` dan `Movie` simpul dan `Rated` tepi. Menggunakan model rekomendasi prediksi tautan Neptunus ML yang sesuai, Anda dapat menambahkan pengguna baru ke grafik dan meminta model memprediksi film untuk mereka, tetapi model hanya dapat merekomendasikan film yang ada selama pelatihan model. Meskipun penyematan `User` simpul dihitung secara real-time menggunakan subgraf lokal dan model GNN, dan karena itu dapat berubah seiring waktu saat pengguna menilai film, ini dibandingkan dengan penyematan film statis yang telah dihitung sebelumnya untuk rekomendasi akhir.
- Model KGE yang didukung oleh Neptunus ML hanya berfungsi untuk tugas prediksi tautan, dan representasi khusus untuk simpul dan tipe tepi yang ada dalam grafik selama pelatihan. Ini berarti bahwa semua simpul dan tipe tepi yang dirujuk dalam kueri inferensi harus ada dalam grafik selama pelatihan. Prediksi untuk tipe tepi atau simpul baru tidak dapat dibuat tanpa melatih ulang model.

SageMaker Keterbatasan sumber daya AI

Bergantung pada aktivitas dan penggunaan sumber daya dari waktu ke waktu, Anda mungkin menemukan pesan kesalahan yang mengatakan bahwa Anda telah melebihi kuota (ResourceLimitExceeded). dan Anda perlu meningkatkan sumber daya SageMaker AI Anda, ikuti langkah-langkah dalam prosedur Minta kuota layanan untuk SageMaker sumber daya di halaman ini untuk meminta peningkatan kuota dari Support. AWS

SageMaker Nama sumber daya AI sesuai dengan tahapan Neptunus ML sebagai berikut:

- SageMaker AI `ProcessingJob` digunakan oleh pemrosesan data Neptunus, pelatihan model, dan pekerjaan transformasi model.
- SageMaker AI `HyperParameterTuningJob` digunakan oleh pekerjaan pelatihan model Neptunus.
- SageMaker AI `TrainingJob` digunakan oleh pekerjaan pelatihan model Neptunus.
- SageMaker AI `Endpoint` digunakan oleh titik akhir inferensi Neptunus.

Memantau sumber daya Amazon Neptunus

Amazon Neptunus mendukung berbagai metode untuk memantau kinerja dan penggunaan database:

- Status instans — Periksa kesehatan mesin database grafik klaster Neptune, cari tahu versi mesin yang diinstal, dan dapatkan informasi terkait instans lainnya menggunakan API status instans.
- API ringkasan grafik — API ringkasan grafik memungkinkan Anda dengan cepat mendapatkan pemahaman tingkat tinggi tentang ukuran dan konten data grafik Anda.

  > **Note**
  >
  > Karena API ringkasan grafik bergantung pada statistik DFE, API ini hanya tersedia saat statistik diaktifkan, yang tidak terjadi pada tipe instans T3 dan T4G.

- Amazon CloudWatch — Neptunus secara otomatis mengirim metrik CloudWatch ke dan juga mendukung Alarm. CloudWatch Untuk informasi selengkapnya, lihat the section called “Menggunakan CloudWatch”.
- File log audit— Lihat, unduh, atau tonton file log basis data menggunakan konsol Neptune. Untuk informasi selengkapnya, lihat the section called “Log Audit dengan Neptune”.
- Menerbitkan log ke Amazon CloudWatch Logs — Anda dapat mengonfigurasi cluster DB Neptunus untuk mempublikasikan data log audit ke grup log di Amazon Logs. CloudWatch Dengan CloudWatch Log, Anda dapat melakukan analisis real-time dari data log, digunakan CloudWatch untuk membuat alarm dan melihat metrik, dan menggunakan CloudWatch Log untuk menyimpan catatan log Anda dalam penyimpanan yang sangat tahan lama. Lihat Log Neptunus CloudWatch .
- AWS CloudTrail- Neptunus mendukung pencatatan API menggunakan. CloudTrail Untuk informasi selengkapnya, lihat the section called “Mencatat Panggilan API Neptunus dengan AWS CloudTrail”.
- Langganan notifikasi acara— Berlangganan ke acara Neptune untuk tetap mengetahui apa yang sedang terjadi. Untuk informasi selengkapnya, lihat the section called “Notifikasi peristiwa”.
- Penandaan— Gunakan tanda untuk menambahkan metadata ke sumber daya Neptune Anda dan lacak penggunaan berdasarkan tanda. Untuk informasi selengkapnya, lihat the section called “Menandai sumber daya Neptunus”.

Topik

- Periksa Status Kondisi Instans Neptune

- Memantau Neptunus Menggunakan Amazon CloudWatch
- Menggunakan Log Audit dengan Klaster Amazon Neptune
- Menerbitkan Log Neptunus ke Log Amazon CloudWatch
- Mengaktifkan CloudWatch Log Amazon untuk notebook Neptunus
- Menggunakan pencatatan kueri lambat Amazon Neptunus
- Mencatat Panggilan API Amazon Neptunus dengan AWS CloudTrail
- Menggunakan Notifikasi Acara Neptune
- Menandai sumber daya Amazon Neptunus

Periksa Status Kondisi Instans Neptune

Amazon Neptune menyediakan mekanisme untuk memeriksa status database grafik pada host. Ini juga merupakan cara yang baik untuk mengonfirmasi bahwa Anda dapat terhubung ke sebuah instans.

Untuk memeriksa kesehatan instans dan mendapatkan status cluster DB menggunakan`curl`:

```
curl -G https://your-neptune-endpoint:port/status
```

Atau, dimulai dengan rilis mesin 1.2.1.0.R6, Anda dapat menggunakan perintah CLI berikut sebagai gantinya:

```
aws neptunedata get-engine-status
```

Jika instance sehat, `status` perintah mengembalikan objek JSON dengan bidang berikut:

- **`status`**— Atur ke `"healthy"` jika instance tidak mengalami masalah.

  Jika instance pulih dari crash atau dari reboot dan ada transaksi aktif yang berjalan dari shutdown server terbaru, `status` diatur ke. `"recovery"`
- **`startTime`**— Atur ke waktu UTC di mana proses server saat ini dimulai.
- **`dbEngineVersion`**— Setel ke versi mesin Neptunus yang berjalan di cluster DB Anda.

  Jika versi mesin ini telah di-patch secara manual sejak dirilis, nomor versi diawali dengan `"Patch-"`.

- **`role`**— Setel ke `"reader"` jika instance adalah replika baca, atau `"writer"` jika instance adalah instance utama.
- **`dfeQueryEngine`**— Setel ke `"enabled"` jika mesin DFE sepenuhnya diaktifkan, atau `viaQueryHint` jika mesin DFE hanya digunakan dengan kueri yang memiliki petunjuk kueri yang disetel ke `true` (`viaQueryHint`adalah default). `useDFE`
- **`gremlin`**— Berisi informasi tentang bahasa query Gremlin yang tersedia di cluster Anda. Secara khusus, ini berisi `version` bidang yang menentukan TinkerPop versi saat ini yang digunakan oleh mesin.
- **`sparql`**— Berisi informasi tentang bahasa query SPARQL yang tersedia di cluster Anda. Secara khusus, ini berisi `version` bidang yang menentukan versi SPARQL saat ini yang digunakan oleh mesin.
- **`opencypher`**— Berisi informasi tentang bahasa query OpenCypher yang tersedia di cluster Anda. Secara khusus, ini berisi `version` bidang yang menentukan versi OperCypher saat ini yang digunakan oleh mesin.
- **`labMode`**— Berisi Mode Lab pengaturan yang digunakan oleh mesin. Ini adalah daftar selektif pengaturan Mode Lab dan bukan set lengkap. Konsultasikan grup parameter cluster untuk set lengkap yang digunakan.
- **`rollingBackTrxCount`**— Jika ada transaksi yang digulung kembali, bidang ini diatur ke jumlah transaksi tersebut. Jika tidak ada, bidang tidak muncul sama sekali.
- **`rollingBackTrxEarliestStartTime`**— Atur ke waktu mulai transaksi paling awal yang digulirkan kembali. Jika tidak ada transaksi yang diputar kembali, bidang tidak muncul sama sekali.
- **`features`**— Berisi informasi status tentang fitur yang diaktifkan pada cluster DB Anda:
  - **`lookupCache`** – Status Cache pencarian saat ini. Bidang ini hanya muncul di tipe instans `R5d`, karena tipe tersebut adalah satu-satunya instans di mana cache pencarian dapat ada. Bidang ini adalah objek JSON dalam bentuk:

    ```
    "lookupCache": {
      "status": "current lookup cache status"
    }
    ```

    Pada instans `R5d`:

    - Jika cache pencarian diaktifkan, statusnya terdaftar sebagai `"Available"`.
    - Jika cache pencarian dinonaktifkan, statusnya terdaftar sebagai `"Disabled"`.

  - Jika batas disk telah tercapai pada instans, statusnya terdaftar sebagai `"Read Only Mode - Storage Limit Reached"`.
- **`ResultCache`** – Status Hasil kueri cache saat ini. Bidang ini adalah objek JSON dalam bentuk:

```
"ResultCache": {
  "status": "current results cache status"
}
```

  - Jika cache hasil telah diaktifkan, status terdaftar sebagai`"Available"`.
  - Jika cache dinonaktifkan, status terdaftar sebagai`"Disabled"`.
- **`IAMAuthentication`**— Menentukan apakah atau tidak AWS Identity and Access Management (IAM) otentikasi telah diaktifkan pada cluster DB Anda:
  - Jika autentikasi IAM diaktifkan, status terdaftar sebagai. `"enabled"`
  - Jika autentikasi IAM dinonaktifkan, status terdaftar sebagai. `"disabled"`
- **`Streams`**— Menentukan apakah aliran Neptunus telah diaktifkan pada cluster DB Anda:
  - Jika aliran diaktifkan, status terdaftar sebagai`"enabled"`.
  - Jika aliran dinonaktifkan, status terdaftar sebagai`"disabled"`.
- **`AuditLog`**— Sama dengan `enabled` jika log audit diaktifkan, atau sebaliknya`disabled`.
- **`SlowQueryLogs`**— Sama dengan `info` atau `debug` jika pencatatan kueri lambat diaktifkan, atau sebaliknya. `disabled`
- **`QueryTimeout`**— Nilai, dalam milidetik, dari batas waktu kueri.
- **`settings`**— Pengaturan diterapkan pada instance:
  - **`clusterQueryTimeoutInMs`**— Nilai, dalam milidetik, dari batas waktu kueri, ditetapkan untuk seluruh cluster.
  - **`SlowQueryLogsThreshold`**— Nilai, dalam milidetik, dari batas waktu kueri, ditetapkan untuk seluruh cluster.
- **`serverlessConfiguration`**— Pengaturan tanpa server untuk cluster jika berjalan sebagai tanpa server:
  - **`minCapacity`**— Ukuran terkecil yang dapat menyusut instance tanpa server di cluster DB Anda, di Unit Kapasitas Neptunus (). NCUs
  - **`maxCapacity`**— Ukuran terbesar di mana instance tanpa server di cluster DB Anda dapat tumbuh, di Unit Kapasitas Neptunus (). NCUs

Contoh output dari perintah status instance

Berikut ini adalah contoh dari output dari perintah status instans, (dalam hal ini, jalankan pada instans R5d):

```
{
  'status': 'healthy',
  'startTime': 'Thu Aug 24 21:47:12 UTC 2023',
  'dbEngineVersion': '1.2.1.0.R4',
  'role': 'writer',
  'dfeQueryEngine': 'viaQueryHint',
  'gremlin': {'version': 'tinkerpop-3.6.2'},
  'sparql': {'version': 'sparql-1.1'},
  'opencypher': {'version': 'Neptune-9.0.20190305-1.0'},
  'labMode': {
    'ObjectIndex': 'disabled',
    'ReadWriteConflictDetection': 'enabled'
  },
  'features': {
    'SlowQueryLogs': 'disabled',
    'ResultCache': {'status': 'disabled'},
    'IAMAuthentication': 'disabled',
    'Streams': 'disabled',
    'AuditLog': 'disabled'
  },
  'settings': {
    'clusterQueryTimeoutInMs': '120000',
    'SlowQueryLogsThreshold': '5000'
  },
  'serverlessConfiguration': {
    'minCapacity': '1.0',
    'maxCapacity': '128.0'
  }
}
```

Jika ada masalah dengan instans, perintah status mengembalikan kode kesalahan `HTTP 500`. Jika host tidak terjangkau, permintaan kehabisan waktu. Pastikan bahwa Anda mengakses instans dari dalam virtual private cloud (VPC), dan bahwa grup keamanan Anda memungkinkan Anda mengaksesnya.

Memantau Neptunus Menggunakan Amazon CloudWatch

Amazon Neptunus dan CloudWatch Amazon terintegrasi sehingga Anda dapat mengumpulkan dan menganalisis metrik kinerja. Anda dapat memantau metrik ini menggunakan CloudWatch konsol, AWS Command Line Interface (AWS CLI), atau CloudWatch API.

CloudWatch juga memungkinkan Anda mengatur alarm sehingga Anda dapat diberi tahu jika nilai metrik melanggar ambang batas yang Anda tentukan. Anda bahkan dapat mengatur CloudWatch Acara untuk mengambil tindakan korektif jika terjadi pelanggaran. Untuk informasi selengkapnya tentang penggunaan CloudWatch dan alarm, lihat CloudWatch Dokumentasi.

Topik

- Melihat CloudWatch Data (Konsol)
- Melihat CloudWatch Data (AWS CLI)
- Melihat CloudWatch Data (API)
- Menggunakan CloudWatch untuk memantau kinerja instans DB di Neptunus
- Metrik Neptunus CloudWatch
- Dimensi Neptunus CloudWatch

Melihat CloudWatch Data (Konsol)

Untuk melihat CloudWatch data untuk cluster Neptunus (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Di panel Semua Metrik, pilih Neptunus, lalu pilih Identifier. DBCluster
4. Di panel atas, gulir ke bawah untuk melihat daftar lengkap metrik klaster Anda. Opsi metrik Neptune yang tersedia muncul di daftar Tampilan.

Untuk memilih atau membatalkan pilihan masing-masing metrik, di panel hasil, pilih kotak di samping nama sumber daya dan metrik. Grafik yang menampilkan metrik untuk item yang dipilih muncul di bagian bawah konsol. Untuk mempelajari lebih lanjut tentang CloudWatch grafik, lihat Metrik Grafik di CloudWatch Panduan Pengguna Amazon.

Melihat CloudWatch Data (AWS CLI)

Untuk melihat CloudWatch data untuk cluster Neptunus ()AWS CLI

1. Instal AWS CLI. Lihat Panduan Pengguna AWS Command Line Interface untuk instruksi.
2. Gunakan AWS CLI untuk mengambil informasi. CloudWatch Parameter yang relevan untuk Neptunus tercantum dalam. Metrik Neptunus CloudWatch

   Contoh berikut mengambil CloudWatch metrik untuk jumlah permintaan Gremlin per detik untuk cluster. `gremlin-cluster`

   ```
   <![CDATA[
   aws cloudwatch get-metric-statistics \
       --namespace AWS/Neptune  --metric-name GremlinRequestsPerSec \
       --dimensions Name=DBClusterIdentifier,Value=gremlin-cluster \
       --start-time 2018-03-03T00:00:00Z --end-time 2018-03-04T00:00:00Z \
       --period 60 --statistics=Average
   ]]>
   ```

Melihat CloudWatch Data (API)

CloudWatch juga mendukung `Query` tindakan sehingga Anda dapat meminta informasi secara terprogram. Untuk informasi selengkapnya, lihat dokumentasi CloudWatch Query API dan Referensi Amazon CloudWatch API.

Ketika suatu CloudWatch tindakan memerlukan parameter yang khusus untuk pemantauan Neptunus, `MetricName` seperti, gunakan nilai yang tercantum dalam. Metrik Neptunus CloudWatch

Contoh berikut menunjukkan CloudWatch permintaan tingkat rendah, menggunakan parameter berikut:

- `Statistics.member.1 = Average`
- `Dimensions.member.1 = DBClusterIdentifier=gremlin-cluster`
- `Namespace  = AWS/Neptune`
- `StartTime  = 2013-11-14T00:00:00Z`
- `EndTime  = 2013-11-16T00:00:00Z`
- `Period  = 60`

- `MetricName = GremlinRequestsPerSec`

Inilah yang tampak seperti CloudWatch permintaan itu. Namun, ini hanya untuk menunjukkan bentuk permintaan; Anda harus membangun permintaan Anda sendiri berdasarkan metrik dan jangka waktu Anda.

```
https://monitoring.amazonaws.com/
      ?SignatureVersion=2
      &Action=GremlinRequestsPerSec
      &Version=2010-08-01
      &StartTime=2018-03-03T00:00:00
      &EndTime=2018-03-04T00:00:00
      &Period=60
      &Statistics.member.1=Average
      &Dimensions.member.1=DBClusterIdentifier=gremlin-cluster
      &Namespace=AWS/Neptune
      &MetricName=GremlinRequests
      &Timestamp=2018-03-04T17%3A48%3A21.746Z
      &AWSAccessKeyId=AWS Access Key ID;
      &Signature=signature
```

Menggunakan CloudWatch untuk memantau kinerja instans DB di Neptunus

Anda dapat menggunakan CloudWatch metrik di Neptunus untuk memantau apa yang terjadi pada instans DB Anda dan melacak panjang antrian kueri seperti yang diamati oleh database. Metrik berikut sangat berguna:

- **`CPUUtilization`** – Menunjukkan persentase penggunaan CPU.
- **`VolumeWriteIOPs`**— Menunjukkan jumlah rata-rata I/O penulisan disk ke volume cluster, dilaporkan pada interval 5 menit.
- **`MainRequestQueuePendingRequests`** — Menunjukkan jumlah permintaan yang menunggu di eksekusi yang tertunda antrean input.

Anda juga dapat mengetahui berapa banyak permintaan yang tertunda di server dengan menggunakan Titik akhir status kueri Gremlin dengan parameter `includeWaiting`. Ini akan memberikan status semua kueri yang menunggu.

Indikator berikut dapat membantu Anda menyesuaikan strategi penyediaan dan kueri Neptune Anda untuk meningkatkan efisiensi dan kinerja:

- Latensi yang konsisten, tinggi`CPUUtilization`, tinggi `VolumeWriteIOPs` dan rendah `MainRequestQueuePendingRequests` bersama-sama menunjukkan bahwa server secara aktif terlibat memproses permintaan tulis bersamaan pada tingkat yang berkelanjutan, dengan sedikit I/O menunggu.
- Latensi yang konsisten, `CPUUtilization` rendah, `VolumeWriteIOPs` rendah, dan tidak ada `MainRequestQueuePendingRequests` bersama-sama menunjukkan bahwa Anda memiliki kelebihan kapasitas pada instans DB utama untuk memproses permintaan tulis.
- `CPUUtilization` tinggi dan `VolumeWriteIOPs` tinggi tetapi latency variabel dan `MainRequestQueuePendingRequests` bersama-sama menunjukkan bahwa Anda mengirim lebih banyak pekerjaan daripada yang server proses dalam interval tertentu. Pertimbangkan untuk membuat atau mengubah ukuran permintaan batch untuk melakukan jumlah pekerjaan yang sama dengan and/or penskalaan overhead transaksional yang lebih sedikit pada instance utama hingga meningkatkan jumlah utas kueri yang mampu memproses permintaan tulis secara bersamaan.
- Rendah `CPUUtilization` dengan tinggi `VolumeWriteIOPs` berarti bahwa utas kueri sedang menunggu I/O operasi ke lapisan penyimpanan selesai. Jika Anda melihat latensi variabel dan beberapa peningkatan `MainRequestQueuePendingRequests`, pertimbangkan untuk membuat atau mengubah ukuran permintaan batch agar dapat melakukan jumlah pekerjaan yang sama dengan biaya overhead transaksional yang kurang.

Metrik Neptunus CloudWatch

> **Note**
>
> Amazon Neptunus mengirimkan metrik CloudWatch hanya ketika mereka memiliki nilai bukan nol.
> Untuk semua metrik Neptune lainnya, granularitas agregasi adalah 5 menit.

Topik

- Metrik Neptunus CloudWatch
- CloudWatch Metrik yang Sekarang Tidak Digunakan Lagi di Neptunus

Metrik Neptunus CloudWatch

Tabel berikut mencantumkan CloudWatch metrik yang didukung Neptunus.

> **Note**
>
> Semua metrik kumulatif diatur ulang ke nol setiap kali server restart, baik untuk pemeliharaan, reboot, atau pemulihan dari crash.

Metrik Neptunus CloudWatch

Metrik	Deskripsi
`BackupRetentionPeriodStorageUsed`	Jumlah total penyimpanan cadangan, dalam byte, digunakan untuk mendukung jendela penyimpanan cadangan klaster DB Neptune. Termasuk dalam total yang dilaporkan oleh metrik `TotalBackupStorageBilled` .
`BufferCacheHitRatio`	Persentase permintaan yang dilayani oleh cache buffer. Metrik ini dapat berguna dalam mendiagnosis permintaan latensi, karena cache terlewat menginduksi latensi yang signifikan. Jika rasio hit cache di bawah 99,9, pertimban gkan untuk meningkatkan tipe instans untuk meng-cache lebih banyak data dalam memori.
`ClusterReplicaLag`	Untuk replika baca, jumlah lag saat mereplikasi pembaruan dari instans primer, dalam milidetik.
`ClusterReplicaLagMaximum`	Jumlah maksimum lag antara instans primer dan setiap instans DB Neptune dalam klaster DB, dalam milidetik.
`ClusterReplicaLagMinimum`	Jumlah minimum lag antara instans primer dan setiap instans DB Neptune dalam klaster DB, dalam milidetik.

Metrik	Deskripsi
`CPUCreditBalance`	Jumlah kredit CPU yang diakumulasikan oleh satu instans, yang dilaporkan pada interval 5 menit. Anda dapat menggunakan metrik ini untuk menentukan berapa lama instans DB dapat melakukan burst melampaui tingkat performa dasarnya pada tingkat tertentu.
`CPUCreditUsage`	Jumlah CPU yang digunakan selama periode tertentu, yang dilaporkan pada interval 5 menit. Metrik ini mengukur jumlah waktu selama fisik CPUs telah digunakan untuk memproses instruksi oleh virtual yang CPUs dialokasikan ke instans DB.
`CPUSurplusCreditBalance`	Jumlah kredit surplus yang telah digunakan oleh sebuah instans tak terbatas ketika nilai `CPUCreditBalance` -nya nol. Nilai `CPUSurplusCreditBalance` ditutupi dengan kredit CPU yang diperoleh. Jika jumlah kredit surplus melebihi jumlah kredit maksimum yang dapat diperoleh instans dalam jangka waktu 24 jam, kredit surplus yang digunakan di atas jumlah maksimum akan dikenai biaya tambahan. Metrik kredit CPU hanya tersedia dengan frekuensi 5 menit.
`CPUSurplusCreditsCharged`	Jumlah kredit surplus yang dibelanjakan yang tidak dibayar oleh kredit CPU yang diperoleh, dan dikenakan biaya tambahan.
`CPUUtilization`	Persentase penggunaan CPU.
`EngineUptime`	Jumlah waktu, dalam detik, instans telah berjalan.

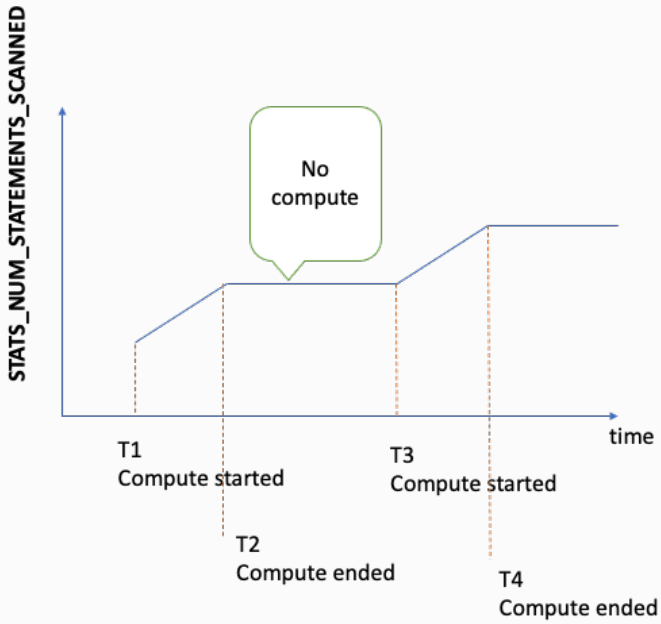
Metrik	Deskripsi
FreeableMemory	Jumlah memori akses acak yang tersedia, dalam byte.
GlobalDbDataTransferBytes	Jumlah byte data redo log yang ditransfer dari primer Wilayah AWS ke sekunder Wilayah AWS dalam database global Neptunus.
GlobalDbReplicatedWriteIO	Jumlah I/O operasi tulis direplikasi dari primer Wilayah AWS dalam database global ke volume cluster di sekunder Wilayah AWS. Perhitungan penagihan untuk setiap cluster DB dalam database global Neptunus menggunak an metrik untuk memperhitungkan penulisan VolumeWriteIOPS yang dilakukan dalam cluster itu. Untuk cluster DB primer, perhitung an penagihan digunakan GlobalDbR eplicatedWriteIO untuk memperhit ungkan replikasi lintas wilayah ke cluster DB sekunder.
GlobalDbProgressLag	Jumlah milidetik yang cluster sekunder berada di belakang cluster utama untuk transaksi pengguna dan transaksi sistem.
GremlinClientErrorsPerSec	Jumlah kesalahan sisi klien per detik di traversal Gremlin.
GremlinServerErrorsPerSec	Jumlah kesalahan sisi server per detik dalam traversal Gremlin.
GremlinRequestsPerSec	Jumlah permintaan per detik untuk mesin Gremlin.
GremlinWebSocketOpenConnections	Jumlah WebSocket koneksi terbuka ke Neptunus.

Metrik	Deskripsi
`LoaderClientErrorsPerSec`	Jumlah kesalahan sisi klien per detik dari permintaan loader.
`LoaderRequestsPerSec`	Jumlah permintaan loader per detik.
`LoaderServerErrorsPerSec`	Jumlah kesalahan sisi server loader per detik.
`MainRequestQueuePendingRequests`	Jumlah permintaan yang menunggu di eksekusi yang tertunda antrean input. Neptune mulai melakukan throttling permintaan ketika mereka melebihi kapasitas antrean maksimum.
`NCUUtilization`	Hanya berlaku untuk instans DB Tanpa Server Neptunus atau cluster DB. Pada tingkat instans, melaporkan persentase yang dihitung sebagai jumlah unit kapasitas Neptunus NCUs () yang saat ini digunakan oleh instance yang dimaksud, dibagi dengan pengaturan kapasitas NCU maksimum untuk cluster. NCU, atau unit kapasitas Neptunus, terdiri dari 2 GiB (gibibyte) memori (RAM), bersama dengan kapasitas prosesor virtual terkait (vCPU) dan jaringan. Pada tingkat cluster, `NCUUtilization` melaporkan persentase kapasitas maksimum yang digunakan oleh cluster secara keseluruh an.
`NetworkThroughput`	Jumlah throughput jaringan yang diterima dari dan dikirim ke klien oleh setiap instans di klaster DB Neptune, dalam byte per detik. Throughput ini tidak termasuk lalu lintas jaringan antara instance di cluster DB dan volume cluster.

Metrik	Deskripsi
NetworkTransmitThroughput	Jumlah throughput jaringan keluar yang ditransmisikan ke klien oleh setiap instance di cluster DB Neptunus, dalam byte per detik. Throughput ini tidak termasuk lalu lintas jaringan antara instance di cluster DB dan volume cluster.
NumIndexDeletesPerSec	Jumlah penghapusan dari indeks individu. Penghapusan dari setiap indeks dihitung secara individual. Ini termasuk penghapusan yang mungkin dibatalkan jika kueri menemukan kesalahan.
NumIndexInsertsPerSec	Jumlah sisipan ke indeks individu. Sisipan untuk setiap indeks dihitung secara terpisah. Ini termasuk sisipan yang mungkin dibatalkan jika kueri menemukan kesalahan.
NumIndexReadsPerSec	Jumlah pernyataan yang dipindai dari indeks apa pun. Setiap pola akses dimulai dengan pencarian pada indeks dan membaca semua pernyataan yang cocok. Peningkatan metrik ini dapat menyebabkan peningkatan latensi kueri atau pemanfaatan CPU.
NumOpenCypherClientErrorsPerSec	Jumlah kesalahan OpenCypher klien per detik.
NumOpenCypherRequestsPerSec	Jumlah OpenCypher permintaan per detik.
NumOpenCypherServerErrorsPerSec	Jumlah kesalahan OpenCypher server per detik.
NumQueuedRequestsPerSec	Jumlah permintaan antri per detik.
NumResultCacheHit	Jumlah hasil cache hasil Gremlin.
NumResultCacheMiss	Jumlah cache hasil Gremlin meleset.

Metrik	Deskripsi
NumTxCommitted	Jumlah transaksi yang berhasil dilakukan per detik.
NumTxOpened	Jumlah transaksi yang dibuka pada server per detik.
NumTxRolledBack	Untuk query tulis, jumlah transaksi per detik digulung kembali di server karena kesalahan . Untuk kueri hanya-baca, metrik ini sama dengan jumlah transaksi hanya-baca yang diselesaikan per detik.
NumUndoPagesPurged	Metrik ini menunjukkan jumlah batch yang dibersihkan. Metrik ini adalah indikator kemajuan dalam pembersihan. Nilainya 0 untuk instance pembaca, dan metrik hanya berlaku untuk instance penulis.
OpenCypherRequestsPerSec	Jumlah permintaan per detik (baik HTTPS dan Bolt) ke mesin OpenCypher.
OpenCypherBoltOpenConnections	Jumlah koneksi Bolt terbuka ke Neptunus.
ResultCacheSizeInBytes	Total perkiraan ukuran (dalam byte) dari semua item yang di-cache di cache hasil Gremlin.
ResultCacheItemCount	Jumlah item dalam cache hasil Gremlin.
ResultCacheOldestItemTimestamp	Stempel waktu item tertua yang di-cache di cache hasil Gremlin.
ResultCacheNewestItemTimestamp	Stempel waktu item terbaru yang di-cache di cache hasil Gremlin.

Metrik	Deskripsi
`ServerlessDatabaseCapacity`	Sebagai metrik tingkat instance, `ServerlessDatabaseCapacity` laporkan kapasitas instance saat ini dari instance tanpa server Neptunus tertentu, di. NCUs NCU, atau unit kapasitas Neptunus, terdiri dari 2 GiB (gibibyte) memori (RAM), bersama dengan kapasitas prosesor virtual terkait (vCPU) dan jaringan. Pada tingkat cluster, `ServerlessDatabaseCapacity` melaporkan rata-rata semua `ServerlessDatabaseCapacity` nilai instans DB di cluster.
`SnapshotStorageUsed`	Jumlah total penyimpanan cadangan yang digunakan oleh semua snapshot untuk klaster Aurora DB Neptune di luar jendela penyimpan an cadangan, dalam byte. Termasuk dalam total yang dilaporkan oleh metrik `TotalBackupStorageBilled` .
`SparqlClientErrorsPerSec`	Jumlah kesalahan sisi klien per detik dalam kueri SPARQL.
`SparqlRequestsPerSec`	Jumlah permintaan per detik untuk mesin SPARQL.
`SparqlServerErrorsPerSec`	Jumlah kesalahan server SPARQL per detik.

Metrik	Deskripsi
`StatsNumStatementsScanned`	Jumlah total pernyataan yang dipindai untuk statistik DFE sejak server dimulai. Setiap kali perhitungan statistik dipicu, angka ini meningkat, tetapi ketika tidak ada perhitung an yang terjadi, itu tetap statis. Akibatnya, jika Anda membuat grafik dari waktu ke waktu, Anda dapat mengetahui kapan perhitungan terjadi dan kapan tidak:  Dengan melihat kemiringan grafik pada periode di mana metrik meningkat, Anda juga dapat mengetahui seberapa cepat perhitungan berjalan. Jika tidak ada metrik seperti itu, itu berarti fitur statistik dinonaktifkan di cluster DB Anda, atau versi mesin yang Anda jalankan tidak memiliki fitur statistik. Jika nilai metrik nol, itu berarti tidak ada perhitungan statistik yang terjadi.

Metrik	Deskripsi
StorageNetworkReceiveThroughput	Jumlah throughput jaringan yang diterima dari subsistem penyimpanan oleh setiap instance di cluster DB Neptunus.
StorageNetworkThroughput	Jumlah throughput jaringan yang diterima dari dan dikirim ke subsistem penyimpanan oleh setiap instance di cluster DB Neptunus.
StorageNetworkTransmitThroughput	Jumlah throughput jaringan yang dikirim ke subsistem penyimpanan oleh setiap instance di cluster DB Neptunus.
SwapUsage	Jumlah ruang swap yang digunakan.
TempStorageIOPS	Jumlah IOPS untuk membaca dan menulis pada penyimpanan lokal yang dilampirkan ke instans DB Neptunus. Metrik ini merepresentasikan hitungan dan diukur sekali per detik.
TempStorageThroughput	Jumlah data yang ditransfer ke dan dari penyimpanan lokal yang terkait dengan instans DB Neptunus. Metrik ini merepresentasikan byte dan diukur sekali per detik.
TotalBackupStorageBilled	Jumlah total penyimpanan cadangan yang akan ditagihkan untuk klaster DB Neptune tertentu, dalam byte. Termasuk penyimpanan backup yang diukur oleh metrik BackupRetentionPeriodStorageUsed dan SnapshotStorageUsed .
TotalRequestsPerSec	Jumlah total permintaan per detik ke server dari semua sumber.

Metrik	Deskripsi
`TotalClientErrorsPerSec`	Jumlah total per detik permintaan yang mengalami kesalahan karena masalah sisi klien.
`TotalServerErrorsPerSec`	Jumlah total per detik permintaan yang mengalami kesalahan pada server karena kegagalan internal.

Metrik	Deskripsi
`UndoLogListSize`	Hitungan batalkan log dalam daftar log undo. Undo log berisi catatan transaksi berkomitm en yang kedaluwarsa ketika semua transaksi aktif lebih baru daripada waktu komit. Catatan yang kedaluwarsa dibersihkan secara berkala. Catatan untuk operasi penghapusan dapat memakan waktu lebih lama untuk dibersihkan daripada catatan untuk jenis transaksi lainnya. Pembersihan dilakukan secara eksklusif oleh instance penulis cluster DB, sehingga tingkat pembersihan tergantung pada jenis instance penulis. Jika tinggi dan berkembang di cluster DB Anda, tingkatkan instance penulis untuk meningkatkan tingkat pembersihan. `UndoLogListSize` Juga, jika Anda meningkatkan ke versi mesin `1.2.0.0` atau lebih tinggi dari versi sebelumny a`1.2.0.0`, pertama-tama pastikan bahwa `UndoLogListSize` nilainya berada di bawah ambang batas tertentu. Jika tidak, tambalan akan berputar kembali dan gagal. Ambang batas didasarkan pada jenis instance: batas default adalah 40k untuk instance 4xlarge atau lebih besar, dan 10k untuk instance yang lebih kecil dari 4xlarge. Jika Anda mencoba memutakhirkan cluster dengan `UndoLogLi stSize` metrik di atas batas, proses tambalan akan kembali, pemutakhiran akan dibatalka n, dan acara dengan alasannya akan terlihat di halaman acara cluster. Batasan ini dapat berubah karena alasan operasional tanpa peringatan sebelumnya. Karena versi engine

Metrik	Deskripsi
	`1.2.0.0` dan yang lebih tinggi menggunak an format yang berbeda untuk membatalkan log, pemutakhiran hanya dapat dimulai setelah log pembatalan sebelumnya telah sepenuhny a dibersihkan di bawah ambang batas yang berlaku. Untuk informasi selengkapnya, lihat Upgrade ke 1.2.0.0 atau lebih tinggi.
`VolumeBytesUsed`	Jumlah total penyimpanan yang dialokasikan untuk klaster DB Neptune Anda, dalam byte. Ini adalah jumlah penyimpanan yang ditagih untuk Anda. Ini adalah jumlah maksimum penyimpan an yang dialokasikan untuk klaster DB Anda pada setiap titik keberadaannya, bukan jumlah yang sedang Anda gunakan (lihat Penagihan penyimpanan Neptunus).
`VolumeReadIOPs`	Jumlah total I/O operasi baca yang ditagih dari volume cluster, melaporkan interval 5 menit. Operasi baca bertagihan dihitung pada tingkat volume klaster, GA dari semua instans dalam klaster DB Neptune, kemudian dilaporkan pada interval 5 menit.
`VolumeWriteIOPs`	Jumlah total I/O operasi tulis disk ke volume cluster, dilaporkan pada interval 5 menit.

CloudWatch Metrik yang Sekarang Tidak Digunakan Lagi di Neptunus

Penggunaan metrik Neptune ini sekarang telah usang. Mereka masih didukung, tetapi dapat dihilangkan di masa depan karena metrik baru dan lebih baik tersedia.

Metrik	Deskripsi
`GremlinHttp1xx`	Jumlah respons HTTP 1xx untuk titik akhir Gremlin per detik. Kami menyarankan agar Anda menggunakan metrik gabungan `Http1xx` baru sebagai gantinya.
`GremlinHttp2xx`	Jumlah respons HTTP 2xx untuk titik akhir Gremlin per detik. Kami menyarankan agar Anda menggunakan metrik gabungan `Http2xx` baru sebagai gantinya.
`GremlinHttp4xx`	Jumlah kesalahan HTTP 4xx untuk titik akhir Gremlin per detik. Kami menyarankan agar Anda menggunakan metrik gabungan `Http4xx` baru sebagai gantinya.
`GremlinHttp5xx`	Jumlah kesalahan HTTP 5xx untuk titik akhir Gremlin per detik. Kami menyarankan agar Anda menggunakan metrik gabungan `Http5xx` baru sebagai gantinya.
`GremlinErrors`	Jumlah kesalahan dalam traversal Gremlin.
`GremlinRequests`	Jumlah permintaan untuk mesin Gremlin.
`GremlinWebSocketSuccess`	Jumlah WebSocket koneksi yang berhasil ke titik akhir Gremlin per detik.
`GremlinWebSocketClientErrors`	Jumlah kesalahan WebSocket klien pada titik akhir Gremlin per detik.

Metrik	Deskripsi
GremlinWebSocketServerErrors	Jumlah kesalahan WebSocket server pada titik akhir Gremlin per detik.
GremlinWebSocketAvailableConnections	Jumlah WebSocket koneksi potensial yang tersedia saat ini.
Http100	Jumlah respons HTTP 100 untuk titik akhir per detik. Kami menyarankan agar Anda menggunakan metrik gabungan Http1xx baru sebagai gantinya.
Http101	Jumlah respons HTTP 101 untuk titik akhir per detik. Kami menyarankan agar Anda menggunakan metrik gabungan Http1xx baru sebagai gantinya.
Http1xx	Jumlah respons HTTP 1xx untuk titik akhir per detik.
Http200	Jumlah respons HTTP 200 untuk titik akhir per detik. Kami menyarankan agar Anda menggunakan metrik gabungan Http2xx baru sebagai gantinya.
Http2xx	Jumlah respons HTTP 2xx untuk titik akhir per detik.

Metrik	Deskripsi
`Http400`	Jumlah kesalahan HTTP 400 untuk titik akhir per detik. Kami menyarankan agar Anda menggunakan metrik gabungan `Http4xx` baru sebagai gantinya.
`Http403`	Jumlah kesalahan HTTP 403 untuk titik akhir per detik. Kami menyarankan agar Anda menggunakan metrik gabungan `Http4xx` baru sebagai gantinya.
`Http405`	Jumlah kesalahan HTTP 405 untuk titik akhir per detik. Kami menyarankan agar Anda menggunakan metrik gabungan `Http4xx` baru sebagai gantinya.
`Http413`	Jumlah kesalahan HTTP 413 untuk titik akhir per detik. Kami menyarankan agar Anda menggunakan metrik gabungan `Http4xx` baru sebagai gantinya.
`Http429`	Jumlah kesalahan HTTP 429 untuk titik akhir per detik. Kami menyarankan agar Anda menggunakan metrik gabungan `Http4xx` baru sebagai gantinya.
`Http4xx`	Jumlah kesalahan HTTP 4xx untuk titik akhir per detik.

Metrik	Deskripsi
`Http500`	Jumlah kesalahan HTTP 500 untuk titik akhir per detik. Kami menyarankan agar Anda menggunak an metrik gabungan `Http5xx` baru sebagai gantinya.
`Http501`	Jumlah kesalahan HTTP 501 untuk titik akhir per detik. Kami menyarankan agar Anda menggunak an metrik gabungan `Http5xx` baru sebagai gantinya.
`Http5xx`	Jumlah kesalahan HTTP 5xx untuk titik akhir per detik.
`LoaderErrors`	Jumlah kesalahan dari permintaan Loader.
`LoaderRequests`	Jumlah Permintaan Loader.
`SparqlHttp1xx`	Jumlah respons HTTP 1xx untuk titik akhir SPARQL per detik. Kami menyarankan agar Anda menggunak an metrik gabungan `Http1xx` baru sebagai gantinya.
`SparqlHttp2xx`	Jumlah respons HTTP 2xx untuk titik akhir SPARQL per detik. Kami menyarankan agar Anda menggunak an metrik gabungan `Http2xx` baru sebagai gantinya.

Metrik	Deskripsi
`SparqlHttp4xx`	Jumlah kesalahan HTTP 4xx untuk titik akhir SPARQL per detik. Kami menyarankan agar Anda menggunakan metrik gabungan `Http4xx` baru sebagai gantinya.
`SparqlHttp5xx`	Jumlah kesalahan HTTP 5xx untuk titik akhir SPARQL per detik. Kami menyarankan agar Anda menggunakan metrik gabungan `Http5xx` baru sebagai gantinya.
`SparqlErrors`	Jumlah kesalahan dalam kueri SPARQL.
`SparqlRequests`	Jumlah permintaan ke mesin SPARQL.
`StatusErrors`	Jumlah kesalahan dari titik akhir status.
`StatusRequests`	Jumlah permintaan ke titik akhir status.

Dimensi Neptunus CloudWatch

Metrik untuk Amazon Neptune dikualifikasikan berdasarkan nilai untuk akun, nama grafik, atau operasi. Anda dapat menggunakan CloudWatch konsol Amazon untuk mengambil data Neptunus bersama dengan salah satu dimensi dalam tabel berikut.

Dimensi	Deskripsi
`DBInstanceIdentifier`	Filter data yang Anda minta untuk instans database tertentu dalam klaster.
`DBClusterIdentifier`	Menyaring data yang Anda minta untuk klaster DB Neptune tertentu.

Dimensi	Deskripsi
`DBClusterIdentifier` , `EngineName`	Memfilter data menurut klaster. Nama mesin untuk semua instans Neptune adalah `neptune`.
`DBClusterIdentifier` , `Role`	Memfilter data yang Anda minta untuk klaster DB Neptune tertentu, menggabungkan metrik menurut peran instans (PENULIS/PEMBACA). Misalnya, Anda dapat mengagregasikan metrik-metrik untuk semua instans PEMBACA milik sebuah klaster.
`DBClusterIdentifier` , `SourceRegion`	Memfilter data oleh cluster utama di wilayah primer database global.
`DatabaseClass`	Filter data yang Anda minta untuk semua instans dalam suatu kelas basis data. Misalnya, Anda dapat menggabungkan metrik untuk semua instans yang termasuk dalam kelas basis data db.r4.large.
`EngineName`	Nama mesin untuk semua instans Neptune adalah `neptune`.
`GlobalDbDBClusterIdentifier` , `SecondaryRegion`	Memfilter data oleh cluster sekunder dari database global tertentu di wilayah sekunder.

Menggunakan Log Audit dengan Klaster Amazon Neptune

Untuk mengaudit aktivitas klaster DB Amazon Neptune, mengaktifkan pengumpulan log audit dengan menetapkan parameter klaster DB. Ketika log audit diaktifkan, Anda dapat menggunakannya untuk mencatat kombinasi peristiwa yang didukung. Anda dapat melihat atau mengunduh log audit untuk meninjaunya.

Mengaktifkan Log Audit Neptune

Gunakan parameter `neptune_enable_audit_log` untuk mengaktifkan (`1`) atau nonaktifkan (`0`) log audit.

Tetapkan parameter ini dalam grup parameter yang digunakan oleh klaster DB Anda. Anda dapat menggunakan prosedur yang ditunjukkan Mengedit Grup Parameter Klaster DB atau Group Parameter DB untuk memodifikasi parameter menggunakan Konsol Manajemen AWS, atau menggunakan perintah modify-db-cluster-parameter-group atau AWS CLI perintah Modify DBCluster ParameterGroup API untuk memodifikasi parameter secara terprogram.

Anda harus me-reboot instance DB Anda setelah memodifikasi parameter ini untuk menerapkan perubahan.

Melihat Log Audit Neptune Menggunakan Konsol

Anda dapat melihat dan mengunduh log Audit menggunakan Konsol Manajemen AWS. Di halaman Instans, pilih instans DB untuk menampilkan detailnya, kemudian gulir ke bagian Log.

Untuk mengunduh file log, pilih file tersebut di bagian Log lalu pilih Unduh.

Detail Log Audit Neptune

File log dalam format UTF-8. Log ditulis di beberapa file, yang jumlahnya bervariasi berdasarkan ukuran instans. Untuk melihat peristiwa terbaru, Anda mungkin harus meninjau semua file log Audit.

Entri log tidak berurutan. Anda dapat menggunakan nilai `timestamp` untuk mengurutkannya.

File log diputar ketika jumlahnya mencapai 100 MB secara keseluruhan. Pembatasan ini tidak dapat dikonfigurasi.

File log audit meliputi informasi yang pisahkan koma berikut dalam baris, dalam urutan berikut:

Bidang	Deskripsi
Stempel waktu	Stempel waktu Unix untuk peristiwa yang dicatat dengan tingkat presisi mikrodetik.
ClientHost	Nama host atau IP yang terhubung ke pengguna.
ServerHost	Nama host atau IP instans yang dicatat untuk peristiwa tersebut.

Bidang	Deskripsi
ConnectionType	Nama koneksi. Bisa`Websocket` ,`HTTP_POST` ,`HTTP_GET`, atau`Bolt`.

ARN IAM Pemanggil	ARN dari pengguna IAM atau IAM role yang digunakan untuk menandatangani permintaan. Kosong jika autentikasi IAM dinonaktifkan. Formatnya adalah:
	`arn:`*`partition`*` :`*`service`*`:`*`region`*`:`*`account`*`:`*`resource`*
	Contoh:
	`arn:aws:iam::123456789012:user/Anna`
	`arn:aws:sts::123456789012:assumed-role/AWSNeptuneNotebookRole/SageMaker`

Konteks Autentikasi	Berisi objek JSON serial yang memiliki informasi autentikasi. Bidang `authenticationSucceeded` adalah `True` jika pengguna diautentikasi. Kosong jika autentikasi IAM dinonaktifkan.
HttpHeader	Informasi header HTTP. Dapat berisi kueri. Kosong untuk WebSocket dan koneksi Baut.
Payload	Kueri Gremlin, SPARQL, atau OpenCypher.

Menerbitkan Log Neptunus ke Log Amazon CloudWatch

Anda dapat mengonfigurasi kluster DB Neptunus untuk mempublikasikan and/or data log audit data log kueri lambat ke grup log di Amazon Logs. CloudWatch Dengan CloudWatch Log, Anda dapat melakukan analisis real-time dari data log, dan menggunakannya CloudWatch untuk membuat alarm dan melihat metrik. Anda dapat menggunakan CloudWatch Log untuk menyimpan catatan log Anda dalam penyimpanan yang sangat tahan lama.

Untuk mempublikasikan log audit ke CloudWatch Log, log audit harus diaktifkan secara eksplisit (lihatMengaktifkan Log Audit). Demikian pula, untuk mempublikasikan log kueri lambat ke CloudWatch Log, log kueri lambat harus diaktifkan secara eksplisit (lihat). Menggunakan pencatatan kueri lambat Amazon Neptunus

> **Note**
>
> Ketahui hal-hal berikut:
>
> - Biaya tambahan berlaku saat Anda mempublikasikan log ke CloudWatch. Lihat halaman CloudWatch harga untuk detailnya.
> - Anda tidak dapat mempublikasikan CloudWatch log ke Log untuk wilayah Tiongkok (Beijing) atau Tiongkok (Ningxia).
> - Jika mengekspor data log dinonaktifkan, Neptune tidak menghapus grup log atau log stream yang ada. Jika mengekspor data log dinonaktifkan, data log yang ada tetap tersedia di CloudWatch Log, tergantung pada penyimpanan log, dan Anda masih dikenakan biaya untuk data log audit yang disimpan. Anda dapat menghapus aliran log dan grup log menggunakan konsol CloudWatch Log, API AWS CLI, atau CloudWatch Logs.

Menggunakan Konsol untuk Mempublikasikan Log Neptunus ke Log CloudWatch

Untuk memublikasikan log Neptunus CloudWatch ke Log dari konsol

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Basis data.
3. Pilih klaster Neptune DB yang data lognya ingin Anda terbitkan.
4. Untuk Tindakan, pilih Ubah.
5. Di bagian Log ekspor, pilih log yang ingin Anda mulai terbitkan ke CloudWatch Log.
6. Pilih Lanjutkan, lalu pilih Modifikasi Klaster DB di halaman ringkasan.

Menggunakan CLI untuk menerbitkan log audit Neptunus ke Log CloudWatch

Anda dapat membuat cluster DB baru yang menerbitkan log audit ke CloudWatch Log menggunakan AWS CLI `create-db-cluster` perintah dengan parameter berikut:

```
aws neptune create-db-cluster \
```

```
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["audit"]'
```

Anda dapat mengonfigurasi cluster DB yang ada untuk mempublikasikan log audit ke CloudWatch Log menggunakan AWS CLI `modify-db-cluster` perintah dengan parameter berikut:

```
aws neptune modify-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Menggunakan CLI untuk menerbitkan log kueri lambat Neptunus ke Log CloudWatch

Anda juga dapat membuat cluster DB baru yang menerbitkan log kueri lambat ke CloudWatch Log menggunakan AWS CLI `create-db-cluster` perintah dengan parameter berikut:

```
aws neptune create-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["slowquery"]'
```

Demikian pula, Anda dapat mengonfigurasi cluster DB yang ada untuk mempublikasikan log kueri lambat ke CloudWatch Log menggunakan AWS CLI `modify-db-cluster` perintah dengan parameter berikut:

```
aws neptune modify-db-cluster --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["slowquery"]}'
```

Memantau Peristiwa Log Neptunus di Amazon CloudWatch

Setelah mengaktifkan log Neptunus, Anda dapat memantau peristiwa log di Amazon Logs. CloudWatch Grup log baru secara otomatis dibuat untuk klaster DB Neptune di bawah awalan berikut, yaitu *`cluster-name`* mewakili nama klaster DB, dan *`log_type`* mewakili tipe log.

```
/aws/neptune/cluster-name/log_type
```

Misalnya, jika Anda mengkonfigurasi fungsi ekspor untuk menyertakan log audit untuk klaster DB bernama `mydbcluster`, data disimpan di grup log `/aws/neptune/mydbcluster/audit`.

Semua peristiwa dari semua instans DB di klaster DB didorong ke grup log menggunakan aliran log yang berbeda.

Jika ada grup log dengan nama yang ditentukan, Neptune menggunakan grup log tersebut untuk mengekspor data log untuk klaster DB Neptune. Anda dapat menggunakan konfigurasi otomatis, seperti AWS CloudFormation, untuk membuat grup log dengan periode penyimpanan log yang telah ditentukan sebelumnya, filter metrik, dan akses pelanggan. Jika tidak, grup log baru secara otomatis dibuat menggunakan periode penyimpanan log default, Never Exire, di CloudWatch Log.

Anda dapat menggunakan konsol CloudWatch Logs, the AWS CLI, atau CloudWatch Logs API untuk mengubah periode penyimpanan log. Untuk informasi selengkapnya tentang mengubah periode penyimpanan CloudWatch log di Log, lihat Mengubah Penyimpanan Data Log di CloudWatch Log.

Anda dapat menggunakan konsol CloudWatch Log, API AWS CLI, atau CloudWatch Logs API untuk mencari informasi dalam peristiwa log untuk klaster DB. Untuk informasi selengkapnya tentang mencari dan memfilter data log, lihat Menelusuri dan Memfilter Data Log.

Mengaktifkan CloudWatch Log Amazon untuk notebook Neptunus

CloudWatch Log untuk notebook Neptunus dinonaktifkan secara default. Ikuti langkah-langkah ini untuk mengaktifkannya, untuk debugging atau tujuan lain:

Menggunakan Konsol Manajemen AWS untuk mengaktifkan CloudWatch Log untuk notebook Neptunus

1. Buka konsol Amazon SageMaker AI di https://console.aws.amazon.com/sagemaker/.
2. Pada panel navigasi di sebelah kiri, pilih Notebook, lalu Instans Notebook. Cari nama notebook Neptunus yang ingin Anda aktifkan log.
3. Buka halaman detail dengan memilih nama instance notebook yang disebutkan pada langkah di atas.
4. Jika instance notebook sedang berjalan, pilih tombol Stop, di kanan atas halaman detail notebook.

5. Di bawah Izin dan enkripsi ada bidang untuk peran IAM ARN. Pilih tautan di bidang ini untuk pergi ke peran IAM untuk buku catatan ini.
6. Buat kebijakan berikut:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DeleteLogDelivery",
        "logs:Describe*",
        "logs:GetLogDelivery",
        "logs:GetLogEvents",
        "logs:ListLogDeliveries",
        "logs:PutLogEvents",
        "logs:PutResourcePolicy",
        "logs:UpdateLogDelivery"
      ],
      "Resource": "*"
    }
  ]
}
```

7. Simpan kebijakan baru ini, dan lampirkan ke peran IAM di langkah 4.
8. Pilih Mulai di kanan atas halaman detail instans notebook SageMaker AI.
9. Setelah log mulai mengalir, Anda akan melihat tautan Lihat Log di bawah bidang berlabel konfigurasi Siklus Hidup di dekat kiri bawah bagian pengaturan instans Notebook pada halaman detail.

Jika notebook Anda gagal memulai, akan ada pesan di halaman detail buku catatan di konsol SageMaker AI yang menyatakan bahwa instance notebook membutuhkan waktu lebih dari 5 menit untuk memulai. CloudWatch Log yang relevan dengan masalah ini dapat ditemukan dengan nama:*(your-notebook-name)*`/LifecycleConfigOnStart`.

Lihat Log SageMaker Acara Amazon dengan Amazon CloudWatch untuk detail selengkapnya, jika perlu.

Menggunakan pencatatan kueri lambat Amazon Neptunus

Mengidentifikasi, men-debug, dan mengoptimalkan kueri yang berjalan lambat bisa jadi sulit. Saat pencatatan kueri lambat Neptunus diaktifkan, atribut dari semua kueri yang berjalan lama secara otomatis dicatat untuk mempermudah proses ini.

> **Note**
>
> Pencatatan kueri lambat diperkenalkan pada rilis mesin Neptunus 1.2.1.0.

Anda mengaktifkan pencatatan kueri lambat menggunakan parameter cluster DB neptune_enable_slow_query_log. Secara default, parameter ini diatur `kedisabled`. Mengaturnya ke `info` atau `debug` mengaktifkan pencatatan kueri lambat. `info`Pengaturan mencatat beberapa atribut berguna dari setiap kueri yang berjalan lambat, sedangkan `debug` pengaturan mencatat semua atribut yang tersedia.

Untuk mengatur ambang batas untuk apa yang dianggap sebagai kueri yang berjalan lambat, gunakan parameter cluster DB neptune_slow_query_log_threshold untuk menentukan jumlah milidetik setelah itu kueri yang berjalan dianggap lambat dan dicatat saat pencatatan kueri lambat diaktifkan. Nilai default adalah 5000 milidetik (5 detik).

Anda dapat mengatur parameter cluster DB ini di Konsol Manajemen AWS, atau menggunakan AWS CLI perintah modify-db-cluster-parameter-group, atau fungsi DBCluster ParameterGroup manajemen Modify.

> **Note**
>
> Parameter pencatatan kueri lambat bersifat dinamis, artinya mengubah nilainya tidak memerlukan atau menyebabkan restart cluster DB Anda.

Untuk melihat log kueri lambat di Konsol Manajemen AWS

Anda dapat melihat dan mengunduh log kueri lambat di Konsol Manajemen AWS, sebagai berikut:

Pada halaman Instances, pilih instans DB dan kemudian gulir ke bagian Log. Anda kemudian dapat memilih file log di sana dan kemudian memilih Unduh untuk mengunduhnya.

File yang dihasilkan oleh pencatatan kueri lambat Neptunus

File log yang dihasilkan oleh pencatatan kueri lambat di Neptunus memiliki karakteristik sebagai berikut:

- File-file tersebut dikodekan sebagai UTF-8.
- Kueri dan atributnya dicatat dalam bentuk JSON.
- Atribut nol dan kosong tidak dicatat, kecuali untuk `queryTime` data.
- Log mencakup beberapa file, yang jumlahnya bervariasi dengan ukuran instance.
- Entri log tidak berurutan. Anda dapat menggunakan `timestamp` nilainya untuk memesannya.
- Untuk melihat peristiwa terbaru, Anda mungkin harus meninjau semua file log kueri lambat.
- File log diputar saat mencapai 100 MiB secara agregat. Pembatasan ini tidak dapat dikonfigurasi.

Atribut kueri masuk **`info`** mode

Atribut berikut dicatat untuk kueri lambat ketika parameter cluster `neptune_enable_slow_query_log` DB telah disetel ke`info`:

Kelompok	Atribut	Deskripsi
requestResponseMetadata	`requestId`	Permintaan id kueri.
	`requestType`	Jenis permintaan, seperti HTTP atau WebSocket.
	`responseStatusCode`	Kode status respons kueri, seperti 200.
	`exceptionClass`	Kelas pengecualian kesalahan dikembalikan setelah eksekusi kueri.
QueryStats	`query`	String kueri.

Kelompok	Atribut	Deskripsi
	`queryFingerprint`	Sidik jari dari kueri.
	`queryLanguage`	Bahasa kueri, seperti Gremlin, SPARQL atau OpenCypher.
MemoryStats	`allocatedPermits`	Izin dialokasikan untuk kueri.
	`approximateUsedMemoryBytes`	Perkiraan memori yang digunakan oleh kueri selama eksekusi.
queryTimeStats	`startTime`	Waktu mulai kueri (UTC).
	`overallRunTimeMs`	Kueri total waktu berjalan, dalam milidetik.
	`parsingTimeMs`	Waktu penguraian kueri, dalam milidetik.
	`waitingTimeMs`	Waktu tunggu Gremlin/SPARQL/openCypher antrian kueri, dalam milidetik
	`executionTimeMs`	Waktu eksekusi kueri, dalam milidetik.
	`serializationTimeMs`	Waktu serialisasi kueri, dalam milidetik.
Penghitung PernyataanPenghitung	`read`	Jumlah pernyataan yang dibaca.
	`written`	Jumlah pernyataan yang ditulis.
	`deleted`	Jumlah pernyataan yang dihapus.

Kelompok	Atribut	Deskripsi
Penghitung Transaksi	`committed`	Jumlah transaksi yang dilakukan.
	`rolledBack`	Jumlah transaksi digulirkan kembali.
VerteXCounters	`added`	Jumlah simpul ditambahkan.
	`removed`	Jumlah simpul dihapus.
	`propertiesAdded`	Jumlah properti simpul ditambahkan.
	`propertiesRemoved`	Jumlah properti simpul dihapus.
EdgeCounters	`added`	Jumlah tepi ditambahkan.
	`removed`	Jumlah tepi dihapus.
	`propertiesAdded`	Jumlah properti tepi ditambahkan.
	`propertiesRemoved`	Jumlah properti tepi dihapus.
ResultCache	`hitCount`	Hasil cache hit count.
	`missCount`	Hasil cache melewatkan hitungan.
	`putCount`	Hasil cache menempatkan hitungan.
concurrentExecutionStats	`acceptedQueryCountAtStart`	Kueri paralel diterima dengan eksekusi kueri saat ini di awal.
	`runningQueryCountAtStart`	Kueri paralel berjalan dengan eksekusi kueri saat ini di awal.

Kelompok	Atribut	Deskripsi
	`acceptedQueryCountAtEnd`	Kueri paralel diterima dengan eksekusi kueri saat ini di akhir.
	`runningQueryCountAtEnd`	Kueri paralel berjalan dengan eksekusi kueri saat ini di akhir.
QueryBatch	`queryProcessingBatchSize`	Ukuran Batch selama pemrosesan kueri.
	`querySerialisationBatchSize`	Ukuran batch selama serialisasi kueri.

Atribut kueri masuk **debug** mode

Ketika parameter cluster `neptune_enable_slow_query_log` DB telah disetel ke`debug`, atribut penghitung penyimpanan berikut dicatat selain atribut yang dicatat seperti dalam `info` mode:

Atribut	Deskripsi
`statementsScannedInAllIndexes`	Pernyataan dipindai di semua indeks.
`statementsScannedSPOGIndex`	Pernyataan dipindai dalam indeks SPOG.
`statementsScannedPOGSIndex`	Pernyataan dipindai dalam indeks POGS.
`statementsScannedGPSOIndex`	Pernyataan dipindai dalam indeks GPSO.
`statementsScannedOSGPIndex`	Pernyataan dipindai dalam indeks OSGP.
`statementsScannedInChunk`	Pernyataan dipindai bersama dalam potongan.
`postFilteredStatementScans`	Pernyataan yang tersisa setelah pemfilteran posting setelah pemindaian.
`distinctStatementScans`	Pernyataan berbeda dipindai.

Atribut	Deskripsi
`statementsReadInAllIndexes`	Pernyataan dibaca setelah pemindaian pemfilteran posting di semua indeks.
`statementsReadSPOGIndex`	Pernyataan dibaca setelah pemindaian posting pemfilteran dalam indeks SPOG.
`statementsReadPOGSIndex`	Pernyataan dibaca setelah pemindaian pemfilteran posting dalam indeks POGS.
`statementsReadGPSOIndex`	Pernyataan dibaca setelah pemindaian posting pemfilteran dalam indeks GPSO.
`statementsReadOSGPIndex`	Pernyataan dibaca setelah pemindaian posting pemfilteran dalam indeks OSGP.
`accessPathSearches`	Jumlah pencarian jalur akses.
`fullyBoundedAccessPathSearches`	Jumlah pencarian jalur akses kunci yang sepenuhnya dibatasi.
`accessPathSearchedByPrefix`	Jumlah jalur akses yang dicari dengan awalan.
`searchesWhereRecordsWereFound`	Jumlah pencarian yang memiliki 1 atau lebih catatan sebagai output.
`searchesWhereRecordsWereNotFound`	Jumlah pencarian yang tidak memiliki catatan sebagai output.
`totalRecordsFoundInSearches`	Total catatan yang ditemukan dari semua pencarian.
`statementsInsertedInAllIndexes`	Jumlah pernyataan yang dimasukkan dalam semua indeks.
`statementsUpdatedInAllIndexes`	Jumlah pernyataan yang diperbarui di semua indeks.

Atribut	Deskripsi
`statementsDeletedInAllIndexes`	Jumlah pernyataan yang dihapus di semua indeks.
`predicateCount`	Jumlah predikat.
`dictionaryReadsFromValueToIdTable`	Jumlah kamus yang dibaca dari nilai ke tabel ID.
`dictionaryReadsFromIdToValueTable`	Jumlah kamus yang dibaca dari ID tabel nilai.
`dictionaryWritesToValueToIdTable`	Jumlah kamus menulis ke nilai ke tabel ID.
`dictionaryWritesToIdToValueTable`	Jumlah kamus menulis ke ID ke tabel nilai.
`rangeCountsInAllIndexes`	Jumlah jumlah rentang di semua indeks.
`deadlockCount`	Jumlah kebuntuan dalam kueri.
`singleCardinalityInserts`	Jumlah sisipan kardinansi tunggal dilakukan.
`singleCardinalityInsertDeletions`	Jumlah pernyataan yang dihapus selama penyisipan kardinalitas tunggal.
`sharedLocksWaitTimeMillis`	Jumlah milidetik yang dihabiskan menunggu kunci bersama.
`exclusiveLocksWaitTimeMillis`	Jumlah milidetik yang dihabiskan menunggu kunci eksklusif.

Contoh logging debug untuk kueri lambat

Kueri Gremlin berikut bisa memakan waktu lebih lama untuk dijalankan daripada ambang batas yang ditetapkan untuk kueri lambat:

```
gremlin=g.V().has('code','AUS').repeat(out().simplePath()).until(has('code','AGR')).path().by('
```

Kemudian, jika pencatatan kueri lambat diaktifkan dalam mode debug, atribut berikut akan dicatat untuk kueri, dalam bentuk seperti ini:


```
{
  "requestResponseMetadata": {
    "requestId": "5311e493-0e98-457e-9131-d250a2ce1e12",
    "requestType": "HTTP_GET",
    "responseStatusCode": 200
  },
  "queryStats": {
    "query":
 "gremlin=g.V().has('code','AUS').repeat(out().simplePath()).until(has('code','AGR')).path().by
    "queryFingerprint":
 "g.V().has(string0,string1).repeat(__.out().simplePath()).until(__.has(string0,string2)).path(
    "queryLanguage": "Gremlin"
  },
  "memoryStats": {
    "allocatedPermits": 20,
    "approximateUsedMemoryBytes": 14838
  },
  "queryTimeStats": {
    "startTime": "23/02/2023 11:42:52.657",
    "overallRunTimeMs": 2249,
    "executionTimeMs": 2229,
    "serializationTimeMs": 13
  },
  "statementCounters": {
    "read": 69979
  },
  "transactionCounters": {
    "committed": 1
  },
  "concurrentExecutionStats": {
    "acceptedQueryCountAtStart": 1
  },
  "queryBatchStats": {
    "queryProcessingBatchSize": 1000,
    "querySerialisationBatchSize": 1000
  },
  "storageCounters": {
    "statementsScannedInAllIndexes": 69979,
    "statementsScannedSPOGIndex": 44936,
    "statementsScannedPOGSIndex": 4,
```

```
    "statementsScannedGPSOIndex": 25039,
    "statementsReadInAllIndexes": 68566,
    "statementsReadSPOGIndex": 43544,
    "statementsReadPOGSIndex": 2,
    "statementsReadGPSOIndex": 25020,
    "accessPathSearches": 27,
    "fullyBoundedAccessPathSearches": 27,
    "dictionaryReadsFromValueToIdTable": 10,
    "dictionaryReadsFromIdToValueTable": 17,
    "rangeCountsInAllIndexes": 4,
    "sharedLocksWaitTimeMillis": 0,
    "exclusiveLocksWaitTimeMillis": 0
  }
}
```

Mencatat Panggilan API Amazon Neptunus dengan AWS CloudTrail

Amazon Neptunus terintegrasi AWS CloudTrail dengan, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Neptunus. CloudTrail menangkap panggilan API untuk Neptunus sebagai peristiwa, termasuk panggilan dari konsol Neptunus dan dari panggilan kode ke Neptunus. APIs

CloudTrail hanya mencatat peristiwa untuk panggilan API Manajemen Neptunus, seperti membuat instance atau cluster. Jika Anda ingin mengaudit perubahan pada grafik Anda, Anda dapat menggunakan log audit. Untuk informasi selengkapnya, lihat Menggunakan Log Audit dengan Klaster Amazon Neptune.

> **Important**
>
> Konsol Amazon Neptune AWS CLI,, dan panggilan API dicatat sebagai panggilan yang dilakukan ke API Amazon Relational Database Service (Amazon RDS).

Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara terus menerus ke bucket Amazon S3, termasuk acara untuk Neptunus. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan

permintaan yang dibuat ke Neptunus, alamat IP dari mana permintaan itu dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

Informasi Neptunus di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas terjadi di Amazon Neptunus, aktivitas tersebut direkam dalam CloudTrail suatu peristiwa bersama dengan peristiwa layanan AWS lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh acara terbaru di AWS akun Anda. Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk acara untuk Neptunus, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol tersebut, jejak diterapkan ke semua Wilayah. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi lebih lanjut, lihat:

- Gambaran Umum untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengkonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah dan Menerima File CloudTrail Log dari Beberapa Akun

Jika tindakan diambil atas nama AWS akun Anda menggunakan konsol Neptunus, antarmuka baris perintah Neptunus, atau SDK Neptunus, mencatat tindakan tersebut sebagai panggilan yang dilakukan ke Amazon APIs RDS API. AWS CloudTrail Misalnya, jika Anda menggunakan konsol Neptunus untuk memodifikasi instans DB atau memanggil perintah, AWS CloudTrail log AWS CLImodify-db-instanceakan menampilkan panggilan ke tindakan Amazon RDS API Modify. DBInstance Untuk daftar tindakan API Neptunus yang dicatat AWS CloudTrail oleh, lihat Referensi API Neptunus.

> **Note**
>
> AWS CloudTrail hanya mencatat peristiwa untuk panggilan API Manajemen Neptunus, seperti membuat instance atau cluster. Jika Anda ingin mengaudit perubahan pada grafik Anda, Anda dapat menggunakan log audit. Untuk informasi selengkapnya, lihat Menggunakan Log Audit dengan Klaster Amazon Neptune.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut ini:

- Apakah permintaan tersebut dibuat dengan kredensial root atau pengguna IAM.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna terfederasi.
- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail .

Memahami entri file log Neptunus

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber mana pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan CloudTrail log untuk pengguna yang membuat snapshot dari instans DB dan kemudian menghapus instance itu menggunakan konsol Neptunus. Konsol diidentifikasi oleh elemen `userAgent`. Panggilan API yang diminta dibuat oleh konsol (`CreateDBSnapshot` dan `DeleteDBInstance`) ditemukan di elemen `eventName` untuk setiap catatan. Informasi tentang pengguna (`Alice`) dapat ditemukan di elemen `userIdentity`.

```
{
  Records:[
  {
    "awsRegion":"us-west-2",
    "eventName":"CreateDBSnapshot",
```

```
    "eventSource":"domainSource",
    "eventTime":"2014-01-14T16:23:49Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-
kaos_fleet-1108-prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"0123456789012",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
    }
  },
  {
    "awsRegion":"us-west-2",
    "eventName":"DeleteDBInstance",
    "eventSource":"domainSource",
    "eventTime":"2014-01-14T16:28:27Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-
kaos_fleet-1108-prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"0123456789012",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
```

```
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
    }
  }
  ]
}
```

Menggunakan Notifikasi Acara Neptune

Topik

- Kategori kejadian dan pesan kejadian Amazon Neptune
- Berlangganan notifikasi peristiwa Neptune
- Mengelola langganan notifikasi peristiwa Neptune

Amazon Neptune menggunakan Amazon Simple Notification Service (Amazon SNS) untuk memberikan pemberitahuan saat kejadian Neptune terjadi. Pemberitahuan ini dapat dalam bentuk apa pun yang didukung oleh Amazon SNS untuk suatu AWS Wilayah, seperti email, pesan teks, atau panggilan ke titik akhir HTTP.

Neptune mengelompokkan kejadian ini ke dalam kategori yang dapat Anda langgani sehingga Anda dapat menerima pemberitahuan saat suatu kejadian dalam kategori tersebut terjadi. Anda dapat berlangganan kategori kejadian untuk instans DB, klaster DB, Snapshot DB, Snapshot klaster DB, atau untuk Grup parameter DB. Misalnya, jika Anda berlangganan kategori Backup untuk instans DB tertentu, Anda akan diberi tahu setiap kali ada kejadian terkait pencadangan yang memengaruhi instans DB. Anda juga menerima pemberitahuan saat langganan pemberitahuan peristiwa berubah.

Peristiwa terjadi di tingkat klaster DB dan instans DB, sehingga Anda dapat menerima peristiwa jika berlangganan ke klaster DB atau instans DB.

Pemberitahuan kejadian dikirim ke alamat yang Anda berikan saat membuat langganan. Anda mungkin ingin membuat beberapa langganan berbeda, seperti satu langganan yang menerima semua notifikasi peristiwa dan langganan lain yang hanya mencakup peristiwa penting untuk instans DB produksi Anda. Anda dapat dengan mudah mematikan notifikasi tanpa menghapus langganan. Untuk melakukannya, atur tombol radio Diaktifkan ke pilihan Tidak di konsol Neptune.

> **Important**
>
> Amazon Neptune tidak menjamin urutan acara yang dikirim dalam aliran kejadian. Urutan peristiwa dapat berubah sewaktu-waktu.

Neptune menggunakan Amazon Resource Name (ARN) dari topik Amazon SNS untuk mengidentifikasi setiap langganan. Konsol Neptune membuat ARN untuk Anda saat Anda membuat langganan.

Penagihan untuk notifikasi acara Neptune adalah melalui Amazon SNS. Biaya Amazon SNS berlaku saat menggunakan pemberitahuan peristiwa. Untuk informasi selengkapnya, lihat Harga Layanan Notifikasi Amazon Simple.

Kategori kejadian dan pesan kejadian Amazon Neptune

Neptune menghasilkan sejumlah besar tindakan dalam beberapa kategori supaya Anda dapat berlangganan dengan menggunakan konsol Neptune. Setiap kategori berlaku untuk jenis sumber, yang dapat berupa instans DB, snapshot DB, atau grup parameter DB.

> **Note**
>
> Neptunus menggunakan definisi acara Amazon RDS yang ada dan. IDs

Peristiwa Neptune yang berasal dari instans DB

Tabel berikut menunjukkan daftar kejadian menurut kategori kejadian saat instans DB merupakan jenis sumber.

Kategori	ID kejadian Amazon RDS	Deskripsi
ketersediaan	RDS-EVENT-0006	Instans DB dimulai ulang.
	RDS-EVENT-0004	Instans DB dimatikan.

Kategori	ID kejadian Amazon RDS	Deskripsi
	RDS-EVENT-0022	Terjadi kesalahan saat me-restart mesin Neptunus.
pencadangan	RDS-EVENT-0001	Mencadangkan instans DB.
	RDS-EVENT-0002	Pencadangan Instans DB Selesai.
perubahan konfigurasi	RDS-EVENT-0009	Instans DB telah ditambahkan ke kelompok keamanan.
	RDS-EVENT-0024	Instans DB sedang dikonversi menjadi instans DB Multi-AZ.
	RDS-EVENT-0030	Instans DB sedang dikonversi menjadi instans DB Single-AZ.
	RDS-EVENT-0012	Menerapkan modifikasi pada kelas instans basis data.
	RDS-EVENT-0018	Pengaturan penyimpanan saat ini untuk instans DB ini diubah.
	RDS-EVENT-0011	Grup parameter untuk instans DB ini telah berubah.

Kategori	ID kejadian Amazon RDS	Deskripsi
	RDS-EVENT-0092	Grup parameter untuk instans DB ini telah selesai diperbarui.
	RDS-EVENT-0028	Pencadangan otomatis untuk instans DB ini telah dinonaktifkan.
	RDS-EVENT-0032	Pencadangan otomatis untuk instans DB ini telah diaktifkan.
	RDS-EVENT-0025	Instans DB telah dikonversi menjadi instans DB Multi-AZ.
	RDS-EVENT-0029	Instans DB telah dikonversi menjadi instans DB Single-AZ.
	RDS-EVENT-0014	Kelas instans DB untuk instan DB ini telah berubah.
	RDS-EVENT-0017	Pengaturan penyimpanan untuk instans DB ini telah berubah.
	RDS-EVENT-0010	Instans DB telah dihapus dari kelompok keamanan.
pembuatan	RDS-EVENT-0005	Instans DB dibuat.

Kategori	ID kejadian Amazon RDS	Deskripsi
penghapusan	RDS-EVENT-0003	Instans DB telah dihapus.
failover	RDS-EVENT-0034	Neptune tidak mencoba failover yang diminta karena failover baru-baru ini terjadi pada instans DB.
	RDS-EVENT-0013	Failover Multi-AZ yang menghasilkan promosi instans siaga telah dimulai.
	RDS-EVENT-0015	Failover Multi-AZ yang menghasilkan promosi instans siaga selesai. Mungkin waktu beberapa menit bagi DNS untuk mentransfer ke instans DB primer baru.
	RDS-EVENT-0065	Instans telah dipulihkan dari failover parsial.
	RDS-EVENT-0049	Failover Multi-AZ telah selesai.
	RDS-EVENT-0050	Aktivasi Multi-AZ telah dimulai setelah pemulihan instans yang berhasil.

Kategori	ID kejadian Amazon RDS	Deskripsi
	RDS-EVENT-0051	Aktivasi Multi-AZ selesai. Basis data Anda seharusnya dapat diakses sekarang.
	RDS-EVENT-0031	Instans DB gagal karena konfigurasi yang tidak kompatibel atau masalah penyimpanan yang mendasari. Mulailah a point-in-time-restore untuk instance DB.
	RDS-EVENT-0036	Instans DB ada di jaringan yang tidak kompatibel. Beberapa subnet yang ditentukan IDs tidak valid atau tidak ada.

Kategori	ID kejadian Amazon RDS	Deskripsi
	RDS-EVENT-0035	Instans DB memiliki parameter yang tidak valid. Misalnya, jika instans DB tidak dapat dimulai karena parameter terkait memori diatur terlalu tinggi untuk kelas instans ini, tindakan pelanggan adalah memodifikasi parameter memori dan mem-boot ulang instans DB.
	RDS-EVENT-0082	Neptune tidak dapat menyalin data cadangan dari bucket Amazon S3. Sepertinya izin Neptune untuk mengakses bucket Amazon S3 tidak dikonfigurasi dengan benar.

Kategori	ID kejadian Amazon RDS	Deskripsi
penyimpanan rendah	RDS-EVENT-0089	Instans DB telah mengonsumsi lebih dari 90% dari penyimpanan yang dialokasikan. Anda dapat memantau ruang penyimpan an untuk instans DB menggunakan metrik Ruang Penyimpanan Kosong.
	RDS-EVENT-0007	Penyimpanan yang dialokasi kan untuk instans DB telah habis. Untuk mengatasi masalah ini, Anda harus mengaloka sikan penyimpan an tambahan untuk instans DB.
pemeliharaan	RDS-EVENT-0026	Pemeliharaan offline instans DB sedang berlangsung. Instans DB saat ini tidak tersedia.
	RDS-EVENT-0027	Pemeliharaan offline instans DB selesai. Instans DB kini tersedia.

Kategori	ID kejadian Amazon RDS	Deskripsi
	RDS-EVENT-0047	Patching instans DB telah selesai.
pemberitahuan	RDS-EVENT-0044	Pemberitahuan yang dikeluarkan operator. Untuk informasi selengkapnya, lihat pesan kejadian.
	RDS-EVENT-0048	Patching dari instans DB telah ditunda.
	RDS-EVENT-0087	Instans DB telah dihentikan.
	RDS-EVENT-0088	Instans DB telah dimulai.
	RDS-EVENT-0154	Instans DB sedang dimulai karena melebihi waktu maksimum penghentian yang diizinkan.
	RDS-EVENT-0158	Instans DB berada dalam status yang tidak dapat ditingkatkan.
	RDS-EVENT-0173	Instans DB telah di-patch.

Kategori	ID kejadian Amazon RDS	Deskripsi
replika baca	RDS-EVENT-0045	Terjadi kesalahan dalam proses replikasi baca. Untuk mengetahui informasi selengkapnya, lihat pesan peristiwa.
	RDS-EVENT-0046	Replika baca telah melanjutkan replikasi . Pesan ini muncul saat Anda pertama kali membuat replika baca, atau sebagai pesan pemantaua n yang mengonfir masi bahwa replikasi berfungsi dengan benar. Jika pesan ini mengikuti pemberita huan RDS-EVENT -0045, replikasi telah dilanjutkan setelah kesalahan atau setelah replikasi dihentikan.
	RDS-EVENT-0057	Replikasi pada replika baca dihentikan.
	RDS-EVENT-0062	Replikasi pada replika baca dihentikan secara manual.
	RDS-EVENT-0063	Replikasi pada replika baca telah direset.

Kategori	ID kejadian Amazon RDS	Deskripsi
pemulihan	RDS-EVENT-0020	Pemulihan instans DB telah dimulai. Waktu pemulihan beragam dengan jumlah data yang akan dipulihkan.
	RDS-EVENT-0021	Pemulihan instans DB selesai.
	RDS-EVENT-0023	Pencadangan manual telah diminta, tetapi Neptune saat ini dalam proses pembuatan snapshot DB. Mengirim permintaan lagi setelah Neptune menyelesaikan snapshot DB.
	RDS-EVENT-0052	Pemulihan instans Multi-AZ telah dimulai. Waktu pemulihan akan beragam dengan jumlah data yang akan dipulihkan.
	RDS-EVENT-0053	Pemulihan instans Multi-AZ selesai.
restorasi	RDS-EVENT-0008	Instans DB telah dipulihkan dari snapshot DB.

Kategori	ID kejadian Amazon RDS	Deskripsi
	RDS-EVENT-0019	Instans DB telah dipulihkan dari point-in-time cadangan.

Peristiwa Neptune yang berasal dari klaster DB

Tabel berikut menunjukkan daftar kejadian menurut kategori kejadian saat klaster DB merupakan jenis sumber.

Kategori	ID peristiwa RDS	Deskripsi
failover	RDS-EVENT-0069	Failover untuk klaster DB telah gagal.
	RDS-EVENT-0070	Failover untuk klaster DB telah dimulai ulang.
	RDS-EVENT-0071	Failover untuk klaster DB telah selesai.
	RDS-EVENT-0072	Failover untuk klaster DB telah dimulai di dalam Availability Zone yang sama.
	RDS-EVENT-0073	Failover untuk klaster DB telah dimulai di Availability Zone.
	RDS-EVENT-0083	Neptune tidak dapat menyalin data cadangan dari bucket Amazon S3. Sepertiny a izin Neptune untuk mengakses bucket

Kategori	ID peristiwa RDS	Deskripsi
		Amazon S3 tidak dikonfigurasi dengan benar.
pemeliharaan	RDS-EVENT-0156	Klaster DB memiliki peningkatan versi minor mesin DB yang tersedia.
pemberitahuan	RDS-EVENT-0076	Migrasi ke kluster DB Neptune gagal.
	RDS-EVENT-0077	Upaya untuk mengonversi tabel dari database sumber ke formulir database gagal selama migrasi ke cluster DB Neptunus.
	RDS-EVENT-0150	Kluster DB berhenti.
	RDS-EVENT-0151	Klaster DB dimulai.
	RDS-EVENT-0152	Perhentian klaster DB gagal.
	RDS-EVENT-0153	Klaster DB sedang dimulai karena melebihi waktu maksimal yang diperbolehkan untuk dihentikan.

Peristiwa Neptune yang berasal dari snapshot klaster DB

Tabel berikut menunjukkan kategori kejadian dan daftar kejadian saat snapshot klaster DB Neptune merupakan jenis sumber.

Kategori	ID peristiwa RDS	Deskripsi
pencadangan	RDS-EVENT-0074	Pembuatan snapshot klaster DB manual telah dimulai.
pencadangan	RDS-EVENT-0075	Snapshot klaster DB manual telah dibuat.
pemberitahuan	RDS-EVENT-0162	Tugas ekspor snapshot klaster DB gagal.
pemberitahuan	RDS-EVENT-0163	Tugas ekspor snapshot klaster DB dibatalkan.
pemberitahuan	RDS-EVENT-0164	Tugas ekspor snapshot klaster DB selesai.
pencadangan	RDS-EVENT-0168	Membuat snapshot klaster otomatis.
pencadangan	RDS-EVENT-0169	Snapshot klaster otomatis dibuat.
pembuatan	RDS-EVENT-0170	Klaster DB dibuat.
penghapusan	RDS-EVENT-0171	Klaster DB dihapus.
pemberitahuan	RDS-EVENT-0172	Menguah nama klaster DB dari [nama klaster DB lama]

Kategori	ID peristiwa RDS	Deskripsi
		menjadi [nama klaster DB baru].

Peristiwa Neptune yang berasal dari grup parameter klaster DB

Tabel berikut menunjukkan kategori kejadian dan daftar kejadian saat grup parameter klaster DB merupakan jenis sumber.

Kategori	ID peristiwa RDS	Deskripsi
perubahan konfigurasi	RDS-EVENT-0037	Grup parameter dimodifikasi.

Peristiwa Neptune yang berasal dari grup keamanan

Tabel berikut menunjukkan kategori kejadian dan daftar kejadian saat grup keamanan merupakan jenis sumber.

Kategori	ID peristiwa RDS	Deskripsi
perubahan konfigurasi	RDS-EVENT-0038	Kelompok keamanan telah dimodifikasi.
kegagalan	RDS-EVENT-0039	Grup keamanan yang dimiliki oleh [pengguna] tidak ada; otorisasi untuk kelompok keamanan telah dicabut.

Berlangganan notifikasi peristiwa Neptune

Anda dapat menggunakan konsol Neptune untuk berlangganan pemberitahuan peristiwa, sebagai berikut:

Untuk berlangganan pemberitahuan kejadian Neptune

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Langganan peristiwa.
3. Di panel Berlangganan kejadian, pilih Buat langganan kejadian.
4. Di kotak dialog Buat langganan kejadian, lakukan hal berikut:
   a. Untuk Nama, masukkan nama untuk langganan pemberitahuan kejadian.
   b. Untuk Kirim pemberitahuan ke, pilih Amazon SNS ARN yang ada untuk topik Amazon SNS, atau pilih buat topik untuk memasukkan nama topik dan daftar penerima.
   c. Untuk Jenis sumber, pilih jenis sumber.
   d. Pilih Ya untuk mengaktifkan langganan. Jika Anda ingin membuat langganan, tetapi belum ingin mengirim pemberitahuan, pilih Tidak.
   e. Tergantung pada jenis sumber yang Anda pilih, tentukan kategori kejadian dan sumber yang Anda inginkan untuk menjadi pangkal notifikasi kejadian.
   f. Pilih Buat.

Mengelola langganan notifikasi peristiwa Neptune

Jika Anda memilih Berlangganan acara di panel navigasi konsol neptune, Anda dapat menampilkan kategori berlangganan dan daftar langganan Anda saat ini.

Anda juga dapat mengubah atau menghapus langganan tertentu.

Mengubah langganan notifikasi peristiwa Neptune

Untuk mengubah langganan notifikasi kejadian Neptune Anda saat ini

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Berlangganan peristiwa. Panel Langganan peristiwa menampilkan semua langganan pemberitahuan peristiwa.
3. Di panel Berlangganan kejadian, pilih langganan yang ingin Anda ubah dan pilih Edit.

4. Buat perubahan pada langganan di bagian Target atau Sumber. Anda dapat menambahkan atau menghapus pengidentifikasi sumber dengan memilih atau membatalkan pilihan pada bagian Sumber.
5. Pilih Edit. Konsol Neptune menunjukkan bahwa langganan sedang dimodifikasi.

Menghapus langganan notifikasi peristiwa Neptune.

Anda dapat menghapus langganan jika sudah tidak membutuhkannya lagi. Semua pelanggan topik tidak akan lagi menerima pemberitahuan peristiwa yang ditentukan oleh langganan.

Untuk menghapus langganan pemberitahuan kejadian Neptune

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Langganan peristiwa.
3. Di panel Langganan Kejadian, pilih langganan yang ingin Anda hapus.
4. Pilih Hapus.
5. Konsol Neptune menunjukkan bahwa langganan sedang dihapus.

Menandai sumber daya Amazon Neptunus

Anda dapat menggunakan tag Neptunus untuk menambahkan metadata ke sumber daya Neptunus Anda. Selain itu, Anda dapat menggunakan tag dengan kebijakan AWS Identity and Access Management (IAM) untuk mengelola akses ke sumber daya Neptunus dan mengontrol tindakan apa yang dapat diterapkan pada sumber daya tersebut. Terakhir, Anda dapat menggunakan tag untuk melacak biaya dengan mengelompokkan pengeluaran untuk sumber daya serupa yang diberi tag.

Semua sumber daya administratif Neptunus dapat ditandai, termasuk yang berikut:

- Instans DB
- Klaster DB
- Replika Baca
- Snapshot DB
- Snapshot klaster DB
- Langganan peristiwa

- Grup parameter DB
- Grup parameter klaster DB
- Grup subnet DB

Ikhtisar tag sumber daya Neptunus

Tag Amazon Neptune adalah pasangan nama dan nilai yang Anda tentukan dan tautkan dengan sumber daya Neptune. Nama ini disebut sebagai kunci. Memberikan nilai untuk kunci bersifat opsional. Anda dapat menggunakan tag untuk memberikan informasi tambahan ke sumber daya Neptune. Anda dapat menggunakan kunci tag, misalnya, untuk menentukan kategori, dan nilai tag mungkin merupakan item dalam kategori tersebut. Misalnya, Anda dapat menentukan kunci tag "proyek" dan nilai tag "Salix", yang menunjukkan bahwa sumber daya Neptune ditugaskan ke proyek Salix. Anda juga dapat menggunakan tag untuk menunjuk sumber daya Neptune sebagai digunakan untuk tes atau produksi dengan menggunakan kunci seperti `environment=test` atau `environment=production`. Kami merekomendasikan bahwa Anda menggunakan serangkaian kunci tag yang konsisten guna mempermudah pelacakan metadata yang terkait dengan sumber daya Neptune.

Gunakan tag untuk mengatur AWS tagihan Anda untuk mencerminkan struktur biaya Anda sendiri. Untuk melakukan ini, daftar untuk mendapatkan Akun AWS tagihan Anda dengan nilai kunci tag disertakan. Kemudian, untuk melihat biaya sumber daya gabungan, atur informasi penagihan Anda sesuai dengan sumber daya Anda dengan nilai kunci tag yang sama. Misalnya, Anda dapat memberi tag pada beberapa sumber daya dengan nama aplikasi tertentu, kemudian mengatur informasi penagihan Anda untuk melihat biaya total aplikasi tersebut di beberapa layanan. Untuk informasi selengkapnya, lihat Menggunakan Tag Alokasi Biaya dalam Panduan Pengguna AWS Billing .

Setiap sumber daya Neptune memiliki rangkaian tag, yang berisi semua tag yang ditetapkan ke sumber daya Neptune tersebut. Rangkaian tanda dapat berisi 10 tanda atau kosong. Jika Anda menambahkan tag ke sumber daya Neptune yang memiliki kunci yang sama dengan tag yang ada pada sumber daya, nilai yang baru akan menimpa nilai yang lama.

AWS tidak menerapkan makna semantik apa pun pada tag Anda; tag ditafsirkan secara ketat sebagai string karakter. Neptune dapat menetapkan tag di instans DB atau sumber daya Neptune lainnya, bergantung pada pengaturan yang Anda gunakan saat membuat sumber daya. Misalnya, Neptune dapat menambahkan tag yang menunjukkan bahwa instans DB digunakan untuk produksi atau untuk pengujian.

- Kunci tanda adalah nama wajib untuk tanda. Nilai string dapat terdiri dari 1 hingga 128 karakter Unicode dan tidak boleh diawali dengan "`aws:`" atau "`rds:`". String hanya dapat berisi kumpulan huruf Unicode, angka, spasi, '_', '.', '/', '=', '+', '-' (Java regex: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`").
- Nilai tag adalah nilai string opsional dari tag. Nilai string dapat terdiri dari 1 hingga 256 karakter Unicode dan tidak boleh diawali dengan "`aws:`". String hanya dapat berisi kumpulan huruf Unicode, angka, spasi, '_', '.', '/', '=', '+', '-' (Java regex: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`").

  Nilai rangkaian tag tidak harus unik dan bisa nol. Misalnya, Anda dapat menggunakan pasangan kunci-nilai dalam satu rangkaian tag `project/Trinity` dan `cost-center/Trinity`.

> **Note**
>
> Anda dapat menambahkan tag ke snapshot. Namun, tagihan Anda tidak akan mencerminkan pengelompokan ini.

Anda dapat menggunakan, API Konsol Manajemen AWS AWS CLI, atau Neptunus untuk menambahkan, membuat daftar, dan menghapus tag pada sumber daya Neptunus. Saat menggunakan AWS CLI atau Neptunus API, Anda harus memberikan Nama Sumber Daya Amazon (ARN) untuk sumber daya Neptunus yang ingin Anda gunakan. Untuk informasi selengkapnya tentang cara membuat konsep ARN, lihat Membangun ARN untuk Neptune.

Tag disimpan di cache untuk diotorisasi. Oleh karena itu, penyediaan tambahan dan pembaruan tag di sumber daya Neptune dapat memakan waktu beberapa menit.

Menyalin tag di Neptunus

Saat membuat atau memulihkan instans DB, Anda dapat menentukan bahwa tag dari instans DB disalin ke snapshot instans DB. Penyalinan tag memastikan bahwa metadata untuk snapshot DB cocok dengan instans DB sumber dan kebijakan akses apa pun untuk snapshot DB juga cocok dengan yang ada di instans DB sumber. Tag tidak disalin secara default.

Anda dapat menentukan bahwa tag disalin ke snapshot DB untuk tindakan berikut:

- Membuat instans DB.
- Memulihkan instans DB.

- Membuat Replika Baca.
- Menyalin snapshot DB.

> **Note**
>
> Jika Anda menyertakan nilai untuk `--tag-key` parameter create-db-cluster-snapshot AWS CLI perintah (atau memberikan setidaknya satu tag ke aksi Buat DBCluster Snapshot API), Neptunus tidak menyalin tag dari instans DB sumber ke snapshot DB baru. Fungsi ini true meskipun instans DB sumber memiliki opsi `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`) yang diaktifkan.
> Ini artinya Anda dapat membuat salinan instans DB dari snapshot DB dan menghindari penambahan tag yang tidak berlaku untuk instans DB baru. Setelah Anda membuat snapshot DB menggunakan AWS CLI `create-db-cluster-snapshot` perintah (atau tindakan `CreateDBClusterSnapshot` Neptunus API), Anda kemudian dapat menambahkan tag seperti yang dijelaskan nanti dalam topik ini.

Menandai di Neptunus menggunakan Konsol Manajemen AWS

Proses pemberian tag pada sumber daya Amazon Neptune untuk semua sumber daya sama. Prosedur berikut menunjukkan cara memberi tag pada instans DB Neptune.

Untuk menambahkan tag ke instans DB

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Instans.

   > **Note**
   >
   > Untuk memfilter daftar instans DB dalam panel Instans, masukkan string teks dalam kotak Filter instans. Hanya instans DB yang berisi string yang muncul.

3. Pilih instans DB yang ingin Anda tandai.
4. Pilih Tindakan Instans, lalu pilih Lihat detail.
5. Di bagian detail, gulir ke bawah hingga bagian Tag.

6. Pilih Tambahkan. Jendela Tambahkan tag akan muncul.
7. Ketik nilai untuk Kunci tag dan Nilai.
8. Untuk menambahkan tag lain, Anda dapat memilih Tambahkan Tag lain dan ketik nilai untuk Kunci tag dan Nilai-nya.

   Ulangi langkah ini seperlunya.
9. Pilih Tambahkan.

Untuk menghapus tag dari instans DB

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Instans.

   > **Note**
   >
   > Untuk memfilter daftar instans DB dalam panel Instans, masukkan string teks dalam kotak Filter instans. Hanya instans DB yang berisi string yang muncul.
3. Pilih instans DB yang ingin Anda tandai.
4. Pilih Tindakan Instans, lalu pilih Lihat detail.
5. Di bagian detail, gulir ke bawah hingga bagian Tag.
6. Pilih tag yang ingin Anda hapus.
7. Pilih Hapus, lalu pilih Hapus di jendela Hapus tanda.

Menandai di Neptunus menggunakan AWS CLI

Anda dapat menambahkan, mencantumkan, atau menghapus tag untuk instans DB di Neptune menggunakan AWS CLI.

- Untuk menambahkan satu atau beberapa tag ke sumber daya Neptunus, gunakan perintah. AWS CLI `add-tags-to-resource`
- Untuk membuat daftar tag pada sumber daya Neptunus, gunakan perintah. AWS CLI `list-tags-for-resource`

- Untuk menghapus satu atau beberapa tag dari sumber daya Neptunus, gunakan perintah. AWS CLI `remove-tags-from-resource`

Untuk mempelajari lebih lanjut tentang cara membuat Amazon Resource Name (ARN) yang diperlukan, lihat Membangun ARN untuk Neptune.

Menandai di Neptunus menggunakan API

Anda dapat menambahkan, mencantumkan, atau menghapus tag untuk instans DB menggunakan API Neptune.

- Untuk menambahkan tag ke sumber daya Neptune, gunakan operasi `AddTagsToResource`.
- Untuk mencantumkan tag yang ditetapkan ke sumber daya Neptune, gunakan `ListTagsForResource`.
- Untuk menghapus tag dari sumber daya Neptune, gunakan operasi `RemoveTagsFromResource`.

Untuk mempelajari lebih lanjut tentang cara membuat konsep ARN yang diperlukan, lihat Membangun ARN untuk Neptune.

Ketika menggunakan XML dengan API Neptune, tag menggunakan skema berikut:

```
<Tagging>
    <TagSet>
        <Tag>
            <Key>Project</Key>
            <Value>Trinity</Value>
        </Tag>
        <Tag>
            <Key>User</Key>
            <Value>Jones</Value>
        </Tag>
    </TagSet>
</Tagging>
```

Tabel berikut menyediakan daftar tag XML yang diizinkan beserta karakteristiknya. Nilai untuk `Key` dan `Value` bergantung pada huruf besar-kecil. Misalnya, `project=Trinity` dan `PROJECT=Trinity` adalah dua tag yang berbeda.

Elemen Penandaan	Deskripsi
`TagSet`	Rangkaian tanda adalah wadah untuk semua tag yang ditetapkan ke sumber daya Neptune. Hanya ada satu rangkaian tag per sumber daya. Anda bekerja dengan `TagSet` hanya melalui API Neptune.
`Tag`	Tag adalah pasangan kunci-nilai yang ditentukan pengguna. Satu rangkaian tag bisa berisi 1 hingga 50 tag.
`Key`	Kunci adalah nama tag yang wajib diisi. Nilai string dapat terdiri dari 1 hingga 128 karakter Unicode dan tidak boleh diawali dengan "`rds:`" atau "`aws:`". String hanya dapat berisi kumpulan huruf Unicode, angka, spasi, '_', '.', '/', '=', '+', '-' (Java regex: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`"). Kunci dalam rangkaian tag harus unik. Misalnya, Anda tidak dapat memiliki pasangan kunci dalam satu rangkaian tag dengan kunci yang sama tetapi dengan nilai yang berbeda, seperti `project/Trinity` dan `project/Xanadu` .
Nilai	Nilai adalah nilai tag opsional. Nilai string dapat terdiri dari 1 hingga 256 karakter Unicode dan tidak boleh diawali dengan "`rds:`" atau "`aws:`". String hanya dapat berisi kumpulan huruf Unicode, angka, spasi, '_', '.', '/', '=', '+', '-' (Java regex: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`"). Nilai rangkaian tag tidak harus unik dan bisa nol. Misalnya, Anda dapat menggunakan pasangan kunci-nilai dalam satu rangkaian tag `project/Trinity` dan `cost-center/Trinity` .

Bekerja dengan administrasi ARNs di Amazon Neptunus

Sumber daya yang dibuat di Amazon Web Services masing-masing diidentifikasi secara unik dengan Amazon Resource Name (ARN). Untuk operasi Amazon Neptune tertentu, Anda harus mengidentifikasi sumber daya Neptune secara unik dengan menentukan ARN-nya.

> **Important**
>
> Amazon Neptunus membagikan format Amazon ARNs RDS untuk tindakan administratif yang menggunakan. Referensi API Manajemen ARNs Administratif `rds` Neptunus mengandung dan tidak. `neptune-db` Untuk bidang data ARNs yang mengidentifikasi sumber daya data Neptunus, lihat Menentukan sumber daya data.

Topik

- Membangun ARN untuk Neptune
- Mendapatkan ARN yang ada di Amazon Neptunus

Membangun ARN untuk Neptune

Anda dapat membangun ARN untuk sumber daya Amazon Neptune menggunakan sintaks berikut. Perhatikan bahwa Neptunus berbagi format Amazon RDS. ARNs

`arn:aws:rds:<region>:<account number>:<resourcetype>:<name>`

Tabel berikut menunjukkan format yang harus Anda gunakan saat membangun ARN untuk jenis sumber daya administratif Neptunus tertentu.

Jenis Sumber Daya	Format ARN
Instans DB	arn:aws:rds:: *<region>* *<account>* :db: *<name>* Contoh: `arn:aws:rds: us-east-2 :123456789012 :db:my-instance-1`
Klaster DB	arn:aws:rds:: *<region>* *<account>* :cluster: *<name>* Contoh: `arn:aws:rds: us-east-2 :123456789012 :cluster: my-cluster-1`
Langganan peristiwa	arn:aws:rds:: *<region>* *<account>* :es: *<name>*

Jenis Sumber Daya	Format ARN
	Contoh: `arn:aws:rds: us-east-2 :123456789012 :es:my-subscription`
Grup parameter DB	arn:aws:rds:: *<region>* *<account>* :pg: *<name>* Contoh: `arn:aws:rds: us-east-2 :123456789012 :pg:my-param-enable-logs`
Grup parameter DB klaster	arn:aws:rds:: *<region>* *<account>* :cluster-pg: *<name>* Contoh: `arn:aws:rds: us-east-2 :123456789012 :cluster-pg: my-cluster-param-timezone`
Snapshot klaster DB	arn:aws:: rds: *<region>* *<account>* :cluster-snapshot: *<name>* Contoh: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-snap-20160809`
Grup subnet DB	arn:aws:: rds: *<region>* *<account>* :subgrp: *<name>* Contoh: `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet-10`

Mendapatkan ARN yang ada di Amazon Neptunus

Anda bisa mendapatkan ARN dari sumber daya Neptunus dengan menggunakan API, Konsol Manajemen AWS(AWS Command Line Interface)AWS CLI, atau Neptunus.

Mendapatkan ARN yang ada menggunakan Konsol Manajemen AWS

Untuk mendapatkan ARN dengan menggunakan konsol, arahkan navigasi ke sumber daya yang Anda inginkan untuk ARN, dan lihat detail untuk sumber daya tersebut. Misalnya, untuk mendapatkan ARN untuk instans DB, pilih Instans di panel navigasi, dan pilih instans yang Anda inginkan dari daftar. ARN berada di bagian Detail Instans.

Mendapatkan ARN yang ada menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk mendapatkan ARN untuk sumber daya Neptunus tertentu, gunakan perintah untuk sumber daya itu. `describe` Tabel berikut menunjukkan setiap AWS CLI perintah dan properti ARN yang digunakan dengan perintah untuk mendapatkan ARN.

AWS CLI Perintah	Properti ARN
describe-event-subscriptions	EventSubscriptionArn
describe-certificates	CertificateArn
describe-db-parameter-groups	DBParameterGroupArn
describe-db-cluster-parameter-kelompok	DBClusterParameterGroupArn
describe-db-instances	DBInstanceArn
describe-events	SourceArn
describe-db-subnet-groups	DBSubnetGroupArn
describe-db-clusters	DBClusterArn
describe-db-cluster-snapshots	DBClusterSnapshotArn

Misalnya, AWS CLI perintah berikut mendapatkan ARN untuk instance DB.

Example

Untuk Linux, OS X, atau Unix:

```
aws neptune describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2
```

Untuk Windows:

```
aws neptune describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2
```

Mendapatkan ARN yang ada menggunakan API

Untuk mendapatkan ARN untuk sumber daya Neptune tertentu, hubungi tindakan API berikut dan gunakan properti ARN yang ditunjukkan.

Tindakan API Neptune	Properti ARN
DescribeEventSubscriptions	EventSubscriptionArn
DescribeCertificates	CertificateArn
Jelaskan DBParameter Grup	DBParameterGroupArn
Jelaskan DBCluster Parameter Groups	DBClusterParameterGroupArn
Jelaskan DBInstances	DBInstanceArn
DescribeEvents	SourceArn
Jelaskan DBSubnet Grup	DBSubnetGroupArn
Jelaskan DBClusters	DBClusterArn
Jelaskan DBCluster Snapshots	DBClusterSnapshotArn

Mencadangkan dan memulihkan cluster DB Amazon Neptunus

Bagian ini menunjukkan cara mencadangkan dan memulihkan klaster DB Amazon Neptune.

Topik

- Cadangan Neptune
- Toleransi kesalahan untuk cluster DB Neptunus
- CloudWatch metrik yang berguna untuk mengelola penyimpanan cadangan Neptunus
- Memulihkan data dari cadangan Neptunus
- Jendela Backup di Neptunus
- Membuat Snapshot Klaster DB di Neptune
- Melakukan pemulihan dari Snapshot Klaster DB
- Menyalin Snapshot Klaster DB
- Menyalin Snapshot Klaster DB
- Menghapus Snapshot Neptune

Cadangan Neptune

Neptune mencadangkan volume klaster Anda secara otomatis dan menyimpan data yang dipulihkan selama periode retensi cadangan. Cadangan Neptune bersifat terus-menerus dan bertahap, sehingga Anda dapat dengan cepat memulihkan ke titik mana pun dalam periode penyimpanan cadangan. Tidak ada dampak performa atau gangguan layanan basis data yang terjadi saat data cadangan ditulis. Anda dapat menentukan periode retensi cadangan, dari 1 hingga 35 hari, saat Anda membuat atau memodifikasi klaster basis data.

Untuk mengontrol penggunaan penyimpanan backup, Anda dapat mengurangi interval retensi backup, menghapus snapshot manual lama saat tidak lagi diperlukan, atau keduanya. Untuk membantu mengelola biaya Anda, Anda dapat memantau jumlah penyimpanan yang digunakan oleh pencadangan berkelanjutan dan snapshot manual yang terus berlanjut melampaui periode penyimpanan. Anda dapat mengurangi interval retensi cadangan dan menghapus snapshot manual ketika tidak diperlukan lagi.

Jika Anda ingin mempertahankan cadangan di luar periode retensi cadangan, Anda juga dapat mengambil cuplikan data itu di volume klaster Anda. Menyimpan snapshot menimbulkan biaya penyimpanan standar untuk Neptune. Untuk informasi selengkapnya tentang harga penyimpanan Neptune, lihat Harga Amazon Neptune.

Neptune menyimpan data pemulihan bertahap selama masa retensi backup. Jadi Anda perlu membuat snapshot untuk data yang ingin Anda simpan melebihi periode retensi cadangan. Anda dapat membuat klaster basis data baru dari cuplikan.

Important

Jika Anda menghapus klaster DB, semua cadangan otomatisnya dihapus pada waktu yang sama dan tidak dapat dipulihkan. Ini berarti kecuali jika Anda memilih untuk membuat snapshot DB akhir secara manual, Anda tidak dapat mengembalikan instans DB ke keadaan akhirnya di lain waktu. Snapshot manual tidak dihapus ketika klaster dihapus.

Note

- Untuk klaster Amazon Neptune, periode retensi cadangan default adalah satu hari bagaimanapun cara klaster DB dibuat.
- Anda tidak dapat menonaktifkan pencadangan otomatis di Neptune. Periode retensi cadangan untuk Neptune dikelola oleh kluster DB.

Toleransi kesalahan untuk cluster DB Neptunus

Klaster DB Neptune didesain agar toleran kesalahan. Volume klaster mencakup beberapa Availability Zone dalam satu AWS Region, dan setiap Availability Zone berisi salinan data volume cluster. Fungsionalitas ini berarti bahwa klaster DB Anda dapat menoleransi kesalahan dari Availability Zone tanpa kehilangan data dan hanya berupa gangguan layanan yang singkat.

Jika instans primer dalam kluster DB gagal, Neptune secara otomatis gagal terhadap satu instans primer baru dalam salah satu dari dua cara:

- Dengan menaikkan replika Neptune yang sudah ada ke instans primer yang baru
- Dengan membuat instans primer baru

Jika klaster DB memiliki satu replika Neptune atau lebih, maka replika Neptune dipromosikan ke instans primer selama peristiwa kegagalan. Peristiwa kegagalan mengakibatkan interupsi singkat, selama operasi baca dan tulis gagal dengan pengecualian. Namun, layanan biasanya dipulihkan dalam waktu kurang dari 120 detik, dan sering kali kurang dari 60 detik. Untuk meningkatkan ketersediaan klaster DB Anda, kami sarankan Anda membuat setidaknya satu replika Neptune atau lebih di dua Availability Zone yang berbeda.

Anda dapat menyesuaikan urutan replika Neptune Anda dinaikkan ke instans primer setelah kegagalan dengan menetapkan masing-masing replika sebagai prioritas. Prioritas berkisar dari 0 untuk prioritas tertinggi hingga 15 untuk prioritas terendah. Jika instans primer gagal, Neptune menaikkan replika Neptune dengan prioritas yang lebih baik untuk instans primer baru. Anda dapat mengubah prioritas dari replika Neptune kapan saja. Memodifikasi prioritas tidak memicu failover.

Anda dapat menggunakan AWS CLI untuk mengatur prioritas failover dari instance DB, sebagai berikut:

```
aws neptune modify-db-instance --db-instance-identifier (the instance ID) --promotion-
tier (the failover priority value)
```

Lebih dari satu replika Neptune dapat memiliki prioritas yang sama, yang menghasilkan tingkat promosi. Jika dua replika Neptune atau lebih memiliki prioritas yang sama, maka Neptune menaikkan replika dengan ukuran paling besar. Jika dua replika Neptune atau lebih memiliki prioritas yang sama, maka Neptune menaikkan replika bebas dengan tingkat promosi yang sama.

Jika klaster DB tidak mengandung replika Neptune, maka instans primer dibuat ulang selama peristiwa kegagalan. Peristiwa kegagalan mengakibatkan gangguan di mana selama operasi baca dan tulis gagal dengan pengecualian. Layanan dipulihkan ketika instans primer baru dibuat, yang biasanya memakan waktu kurang dari 10 menit. Mempromosikan replika Neptune ke instans primer jauh lebih cepat daripada membuat instans primer baru.

CloudWatch metrik yang berguna untuk mengelola penyimpanan cadangan Neptunus

Anda dapat menggunakan CloudWatch metrik `AmazonTotalBackupStorageBilled`,`SnapshotStorageUsed`, dan `BackupRetentionPeriodStorageUsed` untuk meninjau dan memantau jumlah penyimpanan yang digunakan oleh cadangan Neptunus Anda, sebagai berikut:

- `BackupRetentionPeriodStorageUsed` menunjukkan jumlah penyimpanan cadangan yang digunakan, dalam byte, untuk menyimpan pencadangan berkelanjutan pada saat ini. Nilai ini bergantung pada ukuran volume klaster dan jumlah perubahan yang Anda buat selama periode retensi. Namun, untuk tujuan penagihan, nilai tersebut tidak melebihi volume kluster kumulatif selama periode penyimpanan. Misalnya, jika ukuran `VolumeBytesUsed` klaster Anda adalah 107.374.182.400 byte (100 GiB), dan masa penyimpanan Anda adalah dua hari, nilai maksimum untuk `BackupRetentionPeriodStorageUsed` adalah 214.748.364.800 byte (100 GiB + 100 GiB).
- `SnapshotStorageUsed` mewakili jumlah penyimpanan cadangan yang digunakan, dalam byte, untuk menyimpan tangkapan manual di luar periode retensi pencadangan. Snapshot manual tidak dihitung berdasarkan penyimpanan cadangan snapshot Anda sementara stempel waktu pembuatannya dalam periode retensi. Semua snapshot otomatis juga tidak dihitung terhadap pencadangan snapshot. Ukuran masing-masing snapshot adalah ukuran volume klaster pada saat Anda mengambil snapshot. Nilai `SnapshotStorageUsed` bergantung pada jumlah snapshot yang disimpan dan ukuran setiap snapshot. Misalnya, Anda memiliki satu snapshot manual di luar periode retensi, dan ukuran `VolumeBytesUsed` klaster adalah 100 GiB saat snapshot itu diambil. Jumlahnya SnapshotStorageUsed adalah 107.374,182.400 byte (100 GiB).
- `TotalBackupStorageBilled` mewakili jumlah, dalam byte, dari `BackupRetentionPeriodStorageUsed` dan `SnapshotStorageUsed`, dikurangi jumlah penyimpanan cadangan gratis, yang setara dengan ukuran volume kluster selama satu hari. Penyimpanan cadangan gratis setara dengan ukuran volume terbaru. Misalnya, jika ukuran `VolumeBytesUsed` klaster Anda adalah 100 GiB, periode penyimpanan Anda adalah dua hari, dan Anda memiliki satu snapshot manual di luar periode penyimpanan, `TotalBackupStorageBilled` adalah 214.748.364.800 byte (200 GiB + 100 GiB - 100 GiB).

Anda dapat memantau cluster Neptunus dan membuat laporan CloudWatch menggunakan metrik melalui konsol. CloudWatch Untuk informasi selengkapnya tentang cara menggunakan CloudWatch metrik, lihat Pemantauan Neptune dan tabel metrik di. Metrik Neptunus CloudWatch

Memulihkan data dari cadangan Neptunus

Anda dapat memulihkan data Anda dengan membuat klaster DB Neptune baru dari data cadangan yang disimpan Neptune, atau dari snapshot klaster DB yang telah Anda simpan. Anda dapat dengan cepat memulihkan salinan baru klaster DB yang dibuat dari data cadangan ke titik waktu mana pun selama periode retensi cadangan Anda. Keberlangsungan dan inkremental backup Neptune

selama masa retensi pencadangan berarti Anda tidak perlu sering mengambil snapshot data untuk meningkatkan waktu pemulihan.

Untuk menentukan waktu pemulihan terbaru atau paling awal untuk instans DB, cari nilai `Latest Restorable Time` atau `Earliest Restorable Time` pada konsol Neptune. Waktu yang paling baru untuk klaster DB adalah titik terbaru di mana Anda dapat memulihkan klaster DB Anda, biasanya di dalam 5 menit dari waktu saat ini. Waktu paling awal yang dapat dipulihkan menentukan seberapa jauh dalam retensi cadangan di belakang Anda dapat memulihkan volume kluster Anda.

Anda dapat menentukan kapan pemulihan klaster DB selesai dengan memeriksa nilai `Latest Restorable Time` dan `Earliest Restorable Time`. Nilai `Latest Restorable Time` dan `Earliest Restorable Time` mengembalikan nilai NULL hingga operasi pemulihan selesai. Anda tidak dapat meminta operasi pencadangan atau pemulihan jika `Latest Restorable Time` atau `Earliest Restorable Time` mengembalikan NULL.

Untuk mengembalikan instans DB ke waktu tertentu menggunakan Konsol Manajemen AWS

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Instans. Pilih instans primer untuk klaster DB yang ingin Anda pulihkan.
3. Pilih Tindakan instans, lalu pilih Pulihkan ke titik waktu.

   Di jendela Luncurkan instans DB, pilih Kustom di bawah Waktu pemulihan.
4. Tentukan tanggal dan waktu tujuan Anda ingin memulihkan di bawah Kustom.
5. Ketik nama untuk instans DB baru yang dipulihkan untuk Pengidentifikasi instans DB di bawah Pengaturan.
6. Pilih Luncurkan instans DB untuk meluncurkan instans DB yang dipulihkan.

   Sebuah instans DB baru dibuat dengan nama yang Anda tentukan, dan klaster DB baru juga dibuat. Nama klaster DB adalah nama klaster DB baru diikuti oleh `-cluster`. Misalnya, jika nama instans DB yang baru adalah `myrestoreddb`, nama klaster DB barunya adalah `myrestoreddb-cluster`.

Jendela Backup di Neptunus

Pencadangan otomatis terjadi setiap hari selama jendela cadangan yang dipilih. Jika backup memerlukan waktu lebih dari yang dialokasikan ke jendela backup, backup berlanjut setelah jendela

berakhir, hingga selesai. Jendela backup tidak dapat menindih jendela pemeliharaan mingguan untuk instans DB.

Selama jendela pencadangan otomatis, penyimpanan I/O mungkin ditangguhkan sebentar saat proses pencadangan diinisialisasi (biasanya di bawah beberapa detik). Anda mungkin akan mengalami peningkatan latensi selama beberapa menit saat pencadangan dilakukan untuk deployment Multi-AZ.

Jendela pencadangan biasanya dipilih secara acak dari blok waktu delapan jam per Wilayah oleh bidang kontrol Amazon RDS yang mendasari Neptune. Blok waktu untuk setiap wilayah tempat jendela pencadangan default ditetapkan didokumentasikan di bagian Jendela Pencadangan pada Panduan Pengguna Amazon RDS.

Membuat Snapshot Klaster DB di Neptune

Neptune membuat snapshot volume penyimpanan klaster DB Anda, mencadangkan seluruh kluster DB, bukan hanya masing-masing basis data. Saat membuat snapshot klaster DB, Anda perlu mengidentifikasi klaster DB mana yang akan Anda cadangkan. Kemudian beri nama snapshot klaster DB sehingga Anda dapat melakukan proses pemulihan darinya nanti. Jumlah waktu yang diperlukan untuk membuat snapshot klaster DB bervariasi sesuai ukuran basis data Anda. Snapshot mencakup seluruh volume penyimpanan. Jadi ukuran file (seperti file sementara) juga memengaruhi jumlah waktu yang diperlukan untuk membuat snapshot.

Anda dapat membuat snapshot cluster DB menggunakan API Konsol Manajemen AWS, AWS CLI, atau Neptunus.

Menggunakan Konsol untuk Membuat Snapshot Klaster DB

Untuk membuat snapshot klaster DB

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Basis Data.
3. Dalam daftar instans DB, pilih instans primer untuk kluster DB.
4. Pilih Tindakan instans, kemudian pilih Ambil Snapshot.

   Jendela Ambil Snapshot DB akan muncul.
5. Masukkan nama snapshot klaster DB di kotak Nama snapshot.
6. Pilih Ambil Snapshot.

Melakukan pemulihan dari Snapshot Klaster DB

Saat Anda membuat snapshot Amazon Neptunus dari kluster DB, Neptunus membuat snapshot volume penyimpanan cluster, mencadangkan semua datanya dan bukan hanya instance individual. Anda nantinya dapat membuat cluster DB baru dengan memulihkan dari snapshot cluster DB ini. Saat Anda memulihkan cluster DB, Anda memberikan nama snapshot cluster DB untuk dipulihkan, dan kemudian memberikan nama untuk cluster DB baru yang dibuat oleh pemulihan.

Daftar Isi

- Hal-hal yang perlu diingat tentang memulihkan cluster DB Neptunus dari snapshot
  - Anda tidak dapat mengembalikan ke cluster DB yang ada
  - Tidak ada contoh yang dipulihkan
  - Tidak ada grup parameter khusus yang dipulihkan
  - Tidak ada grup keamanan khusus yang dipulihkan
  - Anda tidak dapat memulihkan dari snapshot terenkripsi bersama
  - Cluster DB yang dipulihkan menggunakan penyimpanan sebanyak sebelumnya
- Cara mengembalikan dari snapshot
  - Menggunakan Konsol untuk Melakukan Pemulihan dari Snapshot

Hal-hal yang perlu diingat tentang memulihkan cluster DB Neptunus dari snapshot

Anda tidak dapat mengembalikan ke cluster DB yang ada

Proses pemulihan selalu membuat cluster DB baru, sehingga Anda tidak dapat mengembalikan ke cluster DB yang sudah ada.

Tidak ada contoh yang dipulihkan

Cluster DB baru yang dibuat oleh pemulihan tidak memiliki instance yang terkait dengannya.

Segera setelah pemulihan selesai dan cluster DB baru Anda tersedia, secara eksplisit buat instance yang Anda perlukan. Anda dapat melakukan ini di konsol Neptunus, atau menggunakan API. Buat DBInstance

Tidak ada grup parameter khusus yang dipulihkan

Cluster DB baru yang dibuat oleh pemulihan secara otomatis memiliki grup parameter DB default yang terkait dengannya.

Segera setelah pemulihan selesai dan kluster DB baru Anda tersedia, kaitkan grup parameter DB kustom apa pun yang digunakan instance yang Anda pulihkan. Untuk melakukannya, gunakan perintah Modify pada konsol Neptunus, atau API. Memodifikasi DBInstance

> **Important**
>
> Kami menyarankan Anda menyimpan grup parameter kustom yang digunakan dalam cluster DB yang Anda buat snapshot. Kemudian, ketika Anda memulihkan dari snapshot itu, Anda dapat dengan mudah mengaitkan grup parameter yang benar dengan cluster DB yang dipulihkan.

Tidak ada grup keamanan khusus yang dipulihkan

Cluster DB baru yang dibuat oleh pemulihan secara otomatis memiliki grup keamanan default yang terkait dengannya.

Segera setelah pemulihan selesai dan kluster DB baru Anda tersedia, kaitkan grup keamanan khusus yang digunakan oleh instans yang Anda pulihkan. Untuk melakukannya, gunakan perintah Modify pada konsol Neptunus, atau API. Memodifikasi DBInstance

Anda tidak dapat memulihkan dari snapshot terenkripsi bersama

Anda tidak dapat memulihkan cluster DB dari snapshot cluster DB yang dibagikan dan dienkripsi.

Sebagai gantinya, buat salinan snapshot yang tidak dibagikan dan pulihkan dari salinannya.

Cluster DB yang dipulihkan menggunakan penyimpanan sebanyak sebelumnya

Ketika Anda memulihkan klaster DB dari snapshot klaster DB, jumlah penyimpanan yang dialokasikan untuk klaster baru adalah sama seperti yang dialokasikan untuk klaster DB asal snapshot dibuat, terlepas dari berapa banyak penyimpanan yang dialokasikan sebenarnya digunakan.

Dengan kata lain, “tanda air tinggi” Anda yang ditagih tidak berubah. Mengatur ulang tanda air tinggi memerlukan pengeksporan data dari grafik Anda lalu memuat ulangnya ke klaster DB baru (lihat Penagihan penyimpanan Neptunus).

Cara mengembalikan dari snapshot

Anda dapat memulihkan cluster DB dari snapshot cluster DB menggunakan Konsol Manajemen AWS, API AWS CLI, atau Neptunus.

Menggunakan Konsol untuk Melakukan Pemulihan dari Snapshot

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Snapshot.
3. Pilih snapshot klaster DB yang ingin Anda lakukan pemulihan.
4. Pilih Tindakan, Pulihkan Snapshot.
5. Di halaman Pulihkan Instans DB, di kotak Pengidentifikasi Instans DB, masukkan nama klaster DB Anda yang dipulihkan.
6. Pilih Pulihkan Instans DB.
7. Jika Anda ingin memulihkan fungsionalitas klaster DB ke klaster DB yang menjadi asal pembuatan snapshot, Anda harus memodifikasi klaster DB tersebut untuk menggunakan grup keamanan. Langkah berikutnya menganggap klaster DB Anda berada di Virtual Private Cloud (VPC). Jika klaster DB Anda tidak ada dalam VPC, gunakan konsol Amazon EC2 untuk mencari grup keamanan yang Anda butuhkan untuk klaster DB.

    a. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc/
    b. Pada panel navigasi, pilih Grup Keamanan.
    c. Pilih grup keamanan yang ingin Anda gunakan untuk klaster DB Anda. Jika perlu, tambahkan aturan untuk menghubungkan grup keamanan ke grup keamanan untuk instans EC2.

Menyalin Snapshot Klaster DB

Dengan Neptune, Anda dapat menyalin snapshot klaster DB secara otomatis atau manual. Setelah Anda menyalin snapshot, salinannya disebut snapshot manual.

Anda dapat menyalin snapshot dalam AWS wilayah yang sama dan di seluruh AWS wilayah. Snapshot dapat disalin di seluruh wilayah dan akun dalam satu langkah.

Menyalin snapshot otomatis ke AWS akun lain adalah proses dua langkah: Pertama, Anda membuat snapshot manual dari snapshot otomatis, dan kemudian Anda menyalin snapshot manual ke akun lain.

Sebagai alternatif untuk menyalin, Anda juga dapat berbagi snapshot manual dengan akun lain AWS . Untuk informasi selengkapnya, lihat Menyalin Snapshot Klaster DB.

Topik

- Batasan Menyalin Snapshot
- Retensi Salinan Snapshot Klaster DB
- Menangani Enkripsi Saat Menyalin Snapshots
- Menyalin Snapshot di Seluruh Wilayah AWS
- Menyalin Snapshot Klaster DB Menggunakan Konsol
- Menyalin Snapshot Cluster DB Menggunakan AWS CLI

Batasan Menyalin Snapshot

Berikut adalah beberapa batasan saat Anda menyalin snapshot:

- Anda dapat menyalin snapshot antara Tiongkok (Beijing) dan Tiongkok (Ningxia), tetapi Anda tidak dapat menyalin snapshot antara wilayah Tiongkok ini dan Wilayah lainnya. AWS
- Anda dapat menyalin snapshot antara AWS GovCloud (AS-Timur) dan AWS GovCloud (AS-Barat), tetapi Anda tidak dapat menyalin snapshot antara AWS GovCloud (US) wilayah ini dan Wilayah lainnya. AWS
- Jika Anda menghapus snapshot sumber sebelum snapshot target tersedia, penyalinan snapshot mungkin gagal. Verifikasi bahwa snapshot target telah berstatus `AVAILABLE` sebelum Anda menghapus snapshot sumber.

- Anda dapat memiliki hingga lima snapshot permintaan salinan yang sedang berlangsung ke satu Wilayah per akun.
- Tergantung pada wilayah yang terlibat dan jumlah data yang akan disalin, salinan snapshot lintas wilayah dapat memakan waktu berjam-jam untuk diselesaikan.

  Jika ada sejumlah besar permintaan salinan snapshot lintas wilayah dari Wilayah sumber tertentu, Neptunus dapat menempatkan permintaan salinan lintas AWS wilayah baru dari AWS Wilayah sumber itu ke dalam antrian hingga beberapa salinan yang sedang berlangsung selesai. Tidak ada informasi kemajuan yang ditampilkan tentang permintaan penyalinan saat berada di antrean tersebut. Informasi kemajuan ditampilkan hanya setelah penyalinan dimulai.

Retensi Salinan Snapshot Klaster DB

Neptune menghapus snapshot otomatis sebagai berikut:

- Pada akhir periode retensi snapshot.
- Ketika Anda menonaktifkan snapshot otomatis untuk klaster DB.
- Bila Anda menghapus klaster DB.

Jika Anda ingin mempertahankan snapshot otomatis untuk jangka waktu lebih lama, salin snapshot untuk membuat snapshot manual, yang akan dipertahankan hingga Anda menghapusnya. Biaya penyimpanan Neptune mungkin berlaku untuk snapshot manual jika melebihi ruang penyimpanan default.

Untuk informasi selengkapnya tentang biaya penyimpanan cadangan, lihat Harga Neptune.

Menangani Enkripsi Saat Menyalin Snapshots

Anda dapat menyalin snapshot yang telah dienkripsi menggunakan kunci enkripsi. AWS KMS Jika Anda menyalin snapshot terenkripsi, salinan snapshot tersebut juga harus dienkripsi. Anda dapat mengenkripsi salinan dengan kunci AWS KMS enkripsi yang sama dengan snapshot asli, atau Anda dapat menentukan kunci AWS KMS enkripsi yang berbeda.

Anda tidak dapat mengenkripsi snapshot klaster DB yang tidak dienkripsi saat Anda menyalinnya.

Untuk snapshot cluster Amazon Neptune DB, Anda juga dapat membiarkan snapshot cluster DB tidak terenkripsi dan sebagai gantinya menentukan kunci enkripsi saat memulihkan. AWS KMS Kluster DB yang dipulihkan dienkripsi menggunakan kunci tertentu.

Menyalin Snapshot di Seluruh Wilayah AWS

Saat Anda menyalin snapshot ke AWS Wilayah yang berbeda dari AWS wilayah snapshot sumber, salinan pertama adalah salinan snapshot lengkap, bahkan jika Anda menyalin snapshot tambahan. Salinan snapshot lengkap berisi semua data dan metadata yang diperlukan untuk memulihkan instans DB. Setelah salinan snapshot pertama, Anda dapat menyalin snapshot tambahan dari instans DB yang sama ke wilayah tujuan yang sama dalam akun yang sama. AWS

Snapshot inkremental hanya berisi data yang telah berubah setelah snapshot terbaru dari instans DB yang sama. Penyalinan snapshot inkremental lebih cepat dan menghasilkan biaya penyimpanan yang lebih rendah daripada penyalinan snapshot penuh. Penyalinan snapshot tambahan di seluruh AWS Wilayah didukung untuk snapshot yang tidak terenkripsi dan terenkripsi.

> **Important**
>
> Untuk snapshot yang dibagikan, menyalin snapshot tambahan tidak didukung. Untuk snapshot bersama, semua salinan adalah snapshot penuh, bahkan dalam wilayah yang sama.

Bergantung pada AWS Wilayah yang terlibat dan jumlah data yang akan disalin, salinan snapshot lintas wilayah dapat memakan waktu berjam-jam untuk diselesaikan.

Menyalin Snapshot Klaster DB Menggunakan Konsol

Jika mesin basis data sumber Anda adalah Neptune, snapshot Anda adalah snapshot klaster DB. Untuk setiap AWS akun, Anda dapat menyalin hingga lima snapshot cluster DB sekaligus per AWS Wilayah. Menyalin snapshot klaster DB terenkripsi maupun tak terenkripsi didukung.

Untuk informasi selengkapnya tentang biaya transfer data, lihat Harga Neptune.

Untuk membatalkan operasi penyalinan setelah berlangsung, hapus snapshot klaster DB target saat snapshot klaster DB tersebut berada dalam status menyalin.

Prosedur berikut berfungsi untuk menyalin snapshot klaster DB terenkripsi atau tidak dienkripsi:

Untuk menyalin snapshot klaster DB

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/

2. Di panel navigasi, pilih Snapshot.
3. Pilih kotak centang untuk snapshot klaster DB yang ingin Anda salin.
4. Pilih Tindakan, lalu pilih Salin Snapshot. Halaman Buat Salinan Snapshot DB muncul.
5. Masukkan nama salinan snapshot klaster DB dalam Pengidentifikasi Snapshot DB Baru.
6. Untuk menyalin tanda dan nilai dari snapshot ke salinan snapshot, pilih Salin Tanda.
7. Untuk Aktifkan Enkripsi, pilih salah satu opsi berikut:
   - Pilih Nonaktifkan enkripsi jika snapshot klaster DB tidak dienkripsi dan Anda tidak ingin mengenkripsi salinan tersebut.
   - Pilih Aktifkan enkripsi jika snapshot klaster DB tidak dienkripsi, tetapi Anda ingin mengenkripsi salinan tersebut. Dalam hal ini, untuk Master Key, tentukan pengidentifikasi AWS KMS kunci yang akan digunakan untuk mengenkripsi salinan snapshot cluster DB.
   - Pilih Aktifkan enkripsi jika snapshot klaster DB dienkripsi. Dalam hal ini, Anda harus mengenkripsi salinan, jadi Ya sudah dipilih. Untuk Master Key, tentukan pengidentifikasi AWS KMS kunci yang akan digunakan untuk mengenkripsi salinan snapshot cluster DB.
8. Pilih Salin Snapshot.

Menyalin Snapshot Cluster DB Menggunakan AWS CLI

Anda dapat menyalin snapshot DB menggunakan copy-db-cluster-snapshot AWS CLI perintah.

Jika Anda menyalin snapshot ke AWS Wilayah baru, jalankan perintah di Wilayah baru.

Gunakan deskripsi parameter dan contoh berikut untuk menentukan parameter mana yang akan digunakan dalam menyalin snapshot dengan AWS CLI.

- `--source-db-cluster-snapshot-identifier` – Pengidentifikasi untuk snapshot DB sumber.
  - Jika snapshot sumber berada di AWS Wilayah yang sama dengan salinan, tentukan pengidentifikasi snapshot DB yang valid, seperti. `neptune:instance1-snapshot-20130805`
  - Jika snapshot sumber berada di AWS Wilayah yang berbeda dari salinan, tentukan snapshot DB yang valid seperti ARN. `arn:aws:neptune:us-west-2:123456789012:snapshot:instance1-snapshot-20130805`
  - Jika Anda menyalin dari snapshot DB manual bersama, parameter ini harus merupakan Amazon Resource Name (ARN) dari snapshot DB yang telah dibagikan.

  - Jika Anda menyalin snapshot terenkripsi, parameter ini harus dalam format ARN untuk AWS Wilayah sumber, dan harus cocok dengan parameter. `SourceDBSnapshotIdentifier` `PreSignedUrl`
- `--target-db-cluster-snapshot-identifier` – – Pengidentifikasi untuk salinan baru snapshot DB terenkripsi.
- `--kms-key-id`— — ID AWS KMS kunci untuk snapshot DB terenkripsi. ID AWS KMS kunci adalah Nama Sumber Daya Amazon (ARN), pengenal AWS KMS kunci, atau alias kunci untuk AWS KMS kunci enkripsi. AWS KMS
  - Jika Anda menyalin snapshot DB terenkripsi dari AWS akun Anda, Anda dapat menentukan nilai untuk parameter ini untuk mengenkripsi salinan dengan kunci enkripsi baru. AWS KMS Jika Anda tidak menentukan nilai untuk parameter ini, maka salinan snapshot DB dienkripsi dengan AWS KMS kunci yang sama dengan snapshot DB sumber.
  - Anda tidak dapat menggunakan parameter ini untuk membuat salinan terenkripsi snapshot yang tidak dienkripsi. Mencoba untuk melakukannya akan menghasilkan kesalahan.
  - Jika Anda menyalin snapshot terenkripsi ke AWS Wilayah lain, maka Anda harus menentukan AWS KMS kunci untuk Wilayah tujuan. AWS AWS KMS kunci enkripsi khusus untuk AWS Wilayah tempat mereka dibuat, dan Anda tidak dapat menggunakan kunci enkripsi dari satu AWS Wilayah di AWS Wilayah lain.
- `--source-region`— — ID AWS Wilayah tempat snapshot DB sumber berada. Jika Anda menyalin snapshot terenkripsi ke AWS Wilayah lain, maka Anda harus menentukan opsi ini.
- `--region`— — ID AWS Wilayah tempat Anda menyalin snapshot. Jika Anda menyalin snapshot terenkripsi ke AWS Wilayah lain, maka Anda harus menentukan opsi ini.

Example Dari Tidak Terenkripsi, ke Wilayah yang Sama

Kode berikut membuat salinan snapshot, dengan nama baru`mydbsnapshotcopy`, dari `us-east-1` AWS wilayah ke wilayah `us-west-2` tersebut.

Untuk Linux, OS X, atau Unix:

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

Untuk Windows:

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

Example Dari Tidak Terenkripsi, di Seluruh Wilayah

Kode berikut membuat salinan snapshot, dengan nama barumydbsnapshotcopy, dari us-east-1 AWS wilayah ke wilayah us-west-2 tersebut. Jalankan perintah di wilayah us-west-2.

Untuk Linux, OS X, atau Unix:

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
east-1:123456789012:snapshot:instance1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy \
  --source-region us-east-1 \
  --region us-west-2
```

Untuk Windows:

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
east-1:123456789012:snapshot:instance1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy ^
  --source-region us-east-1 ^
  --region us-west-2
```

Example Dari Terenkripsi, di Seluruh Wilayah

Contoh kode berikut menyalin snapshot DB terenkripsi dari us-east-1 AWS wilayah ke wilayah tersebut. us-west-2 Jalankan perintah di wilayah us-west-2.

Untuk Linux, OS X, atau Unix:

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
west-2:123456789012:snapshot:instance1-snapshot-20161115 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy \
  --source-region us-east-1 \
  --region us-west-2
  --kms-key-id my_us_west_2_key
```

Untuk Windows:

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
west-2:123456789012:snapshot:instance1-snapshot-20161115 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy ^
  --source-region us-east-1 ^
  --region us-west-2
  --kms-key-id my-us-west-2-key
```

Menyalin Snapshot Klaster DB

Dengan Neptune, Anda dapat berbagi snapshot klaster DB manual dengan cara berikut:

- Berbagi snapshot cluster DB manual, baik terenkripsi atau tidak terenkripsi, memungkinkan akun resmi AWS untuk menyalin snapshot.
- Berbagi snapshot cluster DB manual yang tidak terenkripsi memungkinkan AWS akun resmi untuk secara langsung memulihkan cluster DB dari snapshot alih-alih mengambil salinannya dan memulihkannya. Snapshot terenkripsi tidak dapat dipulihkan secara langsung; mereka harus disalin terlebih dahulu dan kemudian dipulihkan dari salinan.

> **Note**
>
> Untuk berbagi snapshot klaster DB otomatis, buat klaster DB manual dengan menyalin snapshot otomatis, lalu membagikannya.

Untuk informasi selengkapnya mengenai pemulihan klaster DB dari snapshot klaster DB, lihat Cara mengembalikan dari snapshot.

Anda dapat membagikan snapshot manual dengan hingga 20 AWS akun lainnya. Anda juga dapat membagikan snapshot manual yang tidak terenkripsi sebagai publik, yang membuat snapshot tersedia untuk semua akun. AWS Berhati-hatilah saat membagikan snapshot sebagai publik sehingga tidak ada informasi pribadi yang dimasukkan ke dalam snapshot publik Anda.

> **Note**
>
> Saat memulihkan kluster DB dari snapshot bersama menggunakan AWS Command Line Interface (AWS CLI) atau Neptunus API, Anda harus menentukan Nama Sumber Daya Amazon (ARN) dari snapshot bersama sebagai pengidentifikasi snapshot.

Topik

- Berbagi Snapshot Klaster DB Terenkripsi
- Menyalin Snapshot Klaster DB

Berbagi Snapshot Klaster DB Terenkripsi

Anda dapat berbagi snapshot klaster DB yang telah dienkripsi "saat istirahat" menggunakan algoritme enkripsi AES-256. Untuk informasi selengkapnya, lihat Mengenkripsi data saat istirahat di database Amazon Neptunus Anda. Untuk melakukannya, Anda harus melakukan langkah-langkah berikut:

1. Bagikan kunci enkripsi AWS Key Management Service (AWS KMS) yang digunakan untuk mengenkripsi snapshot dengan akun apa pun yang Anda inginkan untuk dapat mengakses snapshot.

   Anda dapat berbagi kunci AWS KMS enkripsi dengan AWS akun lain dengan menambahkan akun lain ke kebijakan kunci KMS. Untuk detail tentang pembaruan kebijakan kunci, lihat Kebijakan Kunci dalam Panduan Developer AWS KMS . Untuk contoh pembuatan kebijakan kunci, lihat Membuat Kebijakan IAM untuk Memungkinkan Penyalinan Snapshot Terenkripsi nanti dalam topik ini.
2. Gunakan Konsol Manajemen AWS, AWS CLI, atau Neptunus API untuk membagikan snapshot terenkripsi dengan akun lain.

Batasan ini berlaku untuk berbagi snapshot terenkripsi:

- Anda tidak dapat membagikan foto terenkripsi sebagai publik.
- Anda tidak dapat membagikan snapshot yang telah dienkripsi menggunakan kunci AWS KMS enkripsi default AWS akun yang membagikan snapshot.

Mengizinkan Akses ke Kunci AWS KMS Enkripsi

Agar AWS akun lain dapat menyalin snapshot cluster DB terenkripsi yang dibagikan dari akun Anda, akun yang Anda bagikan snapshot harus memiliki akses ke kunci KMS yang mengenkripsi snapshot. Untuk mengizinkan AWS akun lain mengakses AWS KMS kunci, perbarui kebijakan kunci untuk kunci KMS dengan ARN akun AWS yang Anda bagikan `Principal` sebagai kebijakan kunci KMS. Kemudian izinkan tindakan `kms:CreateGrant`. Lihat Mengizinkan pengguna di akun lain menggunakan kunci KMS di Panduan AWS Key Management Service Pengembang untuk petunjuk umum.

Setelah Anda memberikan akses AWS akun ke kunci enkripsi KMS Anda, untuk menyalin snapshot terenkripsi Anda, AWS akun itu harus membuat pengguna IAM jika belum memilikinya. Pembatasan keamanan KMS tidak mengizinkan penggunaan identitas AWS akun root untuk ini. AWS Akun juga

harus melampirkan kebijakan IAM ke pengguna IAM yang memungkinkan pengguna IAM untuk menyalin snapshot cluster DB terenkripsi menggunakan kunci KMS Anda.

Dalam contoh kebijakan utama berikut, pengguna `111122223333` adalah pemilik kunci enkripsi KMS, dan pengguna `444455556666` adalah akun yang kuncinya digunakan bersama. Kebijakan kunci yang diperbarui ini memberikan akses AWS akun ke kunci KMS dengan menyertakan ARN untuk identitas akun AWS root bagi `444455556666` pengguna `Principal` sebagai kebijakan, dan dengan mengizinkan `kms:CreateGrant` tindakan.

JSON

```
{
    "Id": "key-policy-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::444455556666:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowAttachmentOfPersistentResources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::444455556666:root"
                ]
```

```
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

Membuat Kebijakan IAM untuk Memungkinkan Penyalinan Snapshot Terenkripsi

Setelah AWS akun eksternal memiliki akses ke kunci KMS Anda, pemilik akun tersebut dapat membuat kebijakan yang memungkinkan pengguna IAM yang dibuat untuk akun tersebut menyalin snapshot terenkripsi yang dienkripsi dengan kunci KMS tersebut.

Contoh berikut menunjukkan kebijakan yang dapat dilampirkan ke pengguna IAM untuk `444455556666` akun AWS . Hal ini memungkinkan pengguna IAM menyalin snapshot bersama dari `111122223333` akun AWS yang telah dienkripsi dengan `c989c1dd-a3f2-4a5d-8d96-e793d082ab26` kunci KMS di Wilayah `us-west-2`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey",
```

```
                "kms:CreateGrant",
                "kms:RetireGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"]
        },
        {
            "Sid": "AllowAttachmentOfPersistentResources",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"],
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

Untuk detail tentang pembaruan kebijakan kunci, lihat Kebijakan Kunci dalam Panduan Developer AWS Key Management Service .

Menyalin Snapshot Klaster DB

Anda dapat membagikan snapshot cluster DB menggunakan Konsol Manajemen AWS, API AWS CLI, atau Neptunus.

Menggunakan Konsol untuk Berbagi Snapshot Klaster DB

Dengan konsol Neptune, Anda dapat berbagi snapshot klaster DB manual dengan hingga 20 akun AWS . Anda juga dapat berhenti berbagi snapshot manual dengan satu atau beberapa akun.

Untuk berbagi snapshot klaster DB manual

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/

2. Di panel navigasi, pilih Snapshot.
3. Pilih snapshot manual yang ingin Anda bagikan.
4. Pilih Tindakan, Bagikan Snapshot.
5. Pilih salah satu opsi berikut untuk Visibilitas snapshot DB.

   - Jika sumber tidak terenkripsi, pilih Publik untuk mengizinkan semua akun AWS untuk memulihkan klaster DB dari snapshot klaster DB manual Anda. Atau pilih Private untuk mengizinkan hanya AWS akun yang Anda tentukan untuk memulihkan cluster DB dari snapshot cluster DB manual Anda.

     > ⚠ Warning
     >
     > Jika Anda menyetel visibilitas snapshot DB ke Publik, semua AWS akun dapat memulihkan cluster DB dari snapshot cluster DB manual Anda dan memiliki akses ke data Anda. Jangan berbagi snapshot klaster DB manual apa pun yang berisi informasi pribadi sebagai Publik.

   - Jika sumbernya dienkripsi, Visibilitas snapshot DB ditetapkan sebagai Privat karena snapshot terenkripsi tidak dapat dibagikan sebagai publik.
6. Untuk ID AWS Akun, masukkan pengenal AWS akun untuk akun yang ingin Anda izinkan untuk memulihkan cluster DB dari snapshot manual Anda. Kemudian pilih Tambahkan. Ulangi untuk menyertakan pengenal AWS akun tambahan, hingga 20 AWS akun.

   Jika Anda membuat kesalahan saat menambahkan pengenal AWS akun ke daftar akun yang diizinkan, Anda dapat menghapusnya dari daftar dengan memilih Hapus di sebelah kanan pengenal AWS akun yang salah.
7. Setelah Anda menambahkan pengenal untuk semua AWS akun yang ingin Anda izinkan untuk memulihkan snapshot manual, pilih Simpan.

Untuk berhenti berbagi snapshot cluster DB manual dengan akun AWS

1. Buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Snapshot.
3. Pilih snapshot manual yang ingin Anda hentikan berbagi.
4. Pilih Tindakan, lalu pilih Berbagi Snapshot.

5. Untuk menghapus izin untuk AWS akun, pilih Hapus untuk pengenal AWS akun untuk akun tersebut dari daftar akun resmi.
6. Pilih Simpan.

Menghapus Snapshot Neptune

Anda dapat menghapus snapshot DB menggunakan API manajemen Neptunus AWS CLI, atau Neptunus: Konsol Manajemen AWS

Menghapus Menggunakan Konsol

1. Masuk ke Konsol AWS Manajemen, dan buka konsol Amazon Neptunus di rumah. https://console.aws.amazon.com/neptune/
2. Di panel navigasi, pilih Snapshot.
3. Pilih snapshot DB yang ingin Anda hapus.
4. Untuk Tindakan, pilih Hapus Snapshot.
5. Pilih Hapus di halaman konfirmasi.

Menghapus Menggunakan AWS CLI

Anda juga dapat menghapus snapshot DB menggunakan perintah AWS CLI delete_db_cluster_snapshot, menggunakan parameter `--db-snapshot-identifier` untuk mengidentifikasi snapshot yang ingin Anda hapus:

Untuk Linux, OS X, atau Unix:

```
aws neptune delete-db-cluster-snapshot \
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

Untuk Windows:

```
aws neptune delete-db-cluster-snapshot ^
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

Menghapus Menggunakan API Manajemen Neptune

Anda dapat menggunakan salah satu SDKs untuk menghapus snapshot DB dengan memanggil Hapus DBCluster Snapshot API dan menggunakan `DBSnapshotIdentifier` parameter untuk mengidentifikasi snapshot DB yang akan dihapus.

Praktik terbaik: mendapatkan hasil maksimal dari Neptunus

Berikut adalah beberapa rekomendasi umum untuk bekerja dengan Amazon Neptune. Gunakan informasi ini sebagai referensi untuk segera menemukan rekomendasi untuk menggunakan Amazon Neptune dan memaksimalkan performa.

Daftar Isi

- Pedoman operasional dasar Amazon Neptunus
  - Praktik terbaik keamanan Amazon Neptunus
  - Menghindari kelas instans yang berbeda dalam sebuah klaster
  - Hindari restart berulang selama pemuatan massal
  - Aktifkan Indeks OSGP jika Anda memiliki sejumlah besar predikat
  - Menghindari transaksi yang berjalan lama jika memungkinkan
  - Praktik terbaik untuk menggunakan metrik Neptunus
  - Praktik terbaik untuk menyetel kueri Neptunus
  - Load balancing di seluruh replika baca
  - Memuat lebih cepat menggunakan instance sementara yang lebih besar
  - Mengubah ukuran instans penulis Anda dengan melakukan failover pada replika-baca
  - Coba lagi unggah setelah kesalahan terputus tugas prefetch data
- Praktik Terbaik Umum untuk Menggunakan Gremlin dengan Neptune
  - Konfigurasi Detak Jantung untuk Neptunus Tanpa Server
  - Struktur meningkatkan kueri untuk memanfaatkan mesin DFE
  - Uji kode Gremlin dalam konteks Anda akan menyebarkannya
  - Membuat Penulisan Gremlin Multithreaded yang Efisien
  - Pemangkasan Catatan dengan Properti Waktu Pembuatan
  - Menggunakan Metode datetime( ) untuk Waktu Data Groovy
  - Menggunakan Tanggal dan Waktu Asli untuk Data Waktu GLV
- Praktik terbaik menggunakan klien Gremlin Java dengan Neptunus
  - Gunakan kembali objek klien di beberapa utas
  - Buat objek klien Gremlin Java terpisah untuk titik akhir baca dan tulis
  - Tambahkan beberapa titik akhir replika baca ke kumpulan koneksi Gremlin Java

- Tutup klien untuk menghindari batas koneksi
- Buat koneksi baru setelah failover
- Setel maxInProcessPerConnection dan maxSimultaneousUsagePerConnection ke nilai yang sama
- Kirim kueri ke server sebagai bytecode bukan sebagai string
- Selalu gunakan sepenuhnya ResultSet atau Iterator yang dikembalikan oleh kueri
- Massal menambahkan simpul dan tepi dalam batch
- Menonaktifkan Caching DNS di Mesin Virtual Java
- Secara opsional, atur batas waktu pada tingkat per kueri
- Pecahkan masalah java.util.concurrent.TimeoutException

- Praktik Terbaik Neptunus Menggunakan OpenCypher dan Bolt
  - Buat koneksi baru setelah failover
  - Penanganan koneksi untuk aplikasi berumur panjang
  - Penanganan koneksi untuk AWS Lambda
  - Lebih suka diarahkan ke tepi dua arah dalam kueri
  - Neptunus tidak mendukung beberapa kueri bersamaan dalam suatu transaksi
  - Tutup objek pengemudi saat Anda selesai
  - Gunakan mode transaksi eksplisit untuk membaca dan menulis
    - Transaksi hanya-baca
    - Transaksi mutasi
  - Coba lagi logika untuk pengecualian
  - Mengatur beberapa properti sekaligus menggunakan satu klausa SET
    - Gunakan klausa SET untuk menghapus beberapa properti sekaligus
  - Gunakan kueri berparameter
  - Gunakan peta yang diratakan alih-alih peta bersarang di klausa UNWIND
  - Tempatkan node yang lebih ketat di sisi kiri dalam ekspresi Variable-Length Path (VLP)
  - Hindari pemeriksaan label node yang berlebihan dengan menggunakan nama hubungan granular
  - Tentukan label tepi jika memungkinkan
  - Hindari menggunakan klausa WITH jika memungkinkan
  - Tempatkan filter restriktif sedini mungkin dalam kueri

- Periksa secara eksplisit apakah properti ada
- Jangan gunakan jalur bernama (kecuali jika diperlukan)
- Hindari KUMPULKAN (DISTINCT ())
- Lebih suka fungsi properti daripada pencarian properti individu saat mengambil semua nilai properti
- Lakukan perhitungan statis di luar kueri
- Masukan batch menggunakan UNWIND alih-alih pernyataan individual
- Lebih suka menggunakan kustom IDs untuk node/hubungan
- Hindari melakukan perhitungan ~id dalam kueri
- Memperbarui/Menggabungkan beberapa node

- Praktik Terbaik Neptune Menggunakan SPARQL
  - Mengajukan Kueri Semua Grafik Bernama secara Default
  - Menentukan Grafik Bernama yang Akan Dimuat
  - Memilih Antara FILTER, FILTER...IN, dan VALUES di Kueri Anda

Pedoman operasional dasar Amazon Neptunus

Berikut ini adalah panduan operasional dasar yang harus Anda ikuti saat bekerja dengan Neptune.

- Memahami instans DB Neptune sehingga Anda dapat mengukurnya dengan tepat untuk kinerja dan persyaratan kasus penggunaan. Lihat Klaster dan Instans DB Amazon Neptune.
- Pantau CPU dan penggunaan memori Anda. Hal ini membantu Anda tahu kapan harus bermigrasi ke kelas instans DB dengan kapasitas CPU atau memori yang lebih besar untuk mencapai kinerja kueri yang Anda butuhkan. Anda dapat mengatur Amazon CloudWatch untuk memberi tahu Anda saat pola penggunaan berubah atau saat Anda mendekati kapasitas penerapan. Dengan cara ini, Anda dapat mempertahankan performa dan ketersediaan sistem. Untuk detailnya, lihat Pemantauan instans dan Pemantauan Neptune.

  Karena Neptune memiliki manajer memori sendiri, melihat penggunaan memori yang relatif rendah bahkan ketika penggunaan CPU tinggi itu hal yang normal. Menghadapi out-of-memory pengecualian saat menjalankan kueri adalah indikator terbaik yang Anda butuhkan untuk meningkatkan memori yang dapat dibebaskan.
- Aktifkan pencadangan otomatis dan atur jendela pencadangan agar terjadi pada waktu yang tepat.

- Tes failover untuk instans DB Anda untuk memahami berapa lama proses untuk kasus penggunaan Anda. Hal ini juga membantu memastikan bahwa aplikasi yang mengakses instans DB Anda dapat secara otomatis terhubung ke instans DB baru setelah failover.
- Jika memungkinkan, jalankan klien Anda dan klaster Neptune di wilayah dan VPC yang sama, karena koneksi lintas wilayah dengan peering VPC dapat memperkenalkan penundaan dalam permintaan waktu respons. Untuk respons kueri milidetik satu digit, perlu untuk menjaga klien dan klaster Neptune di wilayah dan VPC yang sama.
- Ketika Anda membuat instance read-replica, itu harus setidaknya sebesar instance penulis utama. Hal ini membantu menjaga kelambatan replikasi di pemeriksaan, dan menghindari replika restart. Lihat Menghindari kelas instans yang berbeda dalam sebuah klaster.
- Sebelum mengupgrade ke versi mesin utama baru, pastikan untuk menguji aplikasi Anda sebelum Anda meng-upgrade. Anda dapat melakukan ini dengan mengkloning klaster DB Anda sehingga klon klaster menjalankan versi mesin baru, lalu menguji aplikasi Anda pada klon.
- Untuk memfasilitasi failover, semua instans idealnya harus berukuran yang sama.

Topik

- Praktik terbaik keamanan Amazon Neptunus
- Menghindari kelas instans yang berbeda dalam sebuah klaster
- Hindari restart berulang selama pemuatan massal
- Aktifkan Indeks OSGP jika Anda memiliki sejumlah besar predikat
- Menghindari transaksi yang berjalan lama jika memungkinkan
- Praktik terbaik untuk menggunakan metrik Neptunus
- Praktik terbaik untuk menyetel kueri Neptunus
- Load balancing di seluruh replika baca
- Memuat lebih cepat menggunakan instance sementara yang lebih besar
- Mengubah ukuran instans penulis Anda dengan melakukan failover pada replika-baca
- Coba lagi unggah setelah kesalahan terputus tugas prefetch data

Praktik terbaik keamanan Amazon Neptunus

Gunakan akun AWS Identity and Access Management (IAM) untuk mengontrol akses ke tindakan Neptunus API. Kontrol tindakan yang membuat, memodifikasi, atau menghapus sumber daya

Neptune (seperti instans DB, grup keamanan, grup opsi, atau grup parameter), dan tindakan yang melakukan tindakan administratif umum (seperti membuat cadangan dan memulihkan instans DB).

- Gunakan kredenal sementara daripada persisten bila memungkinkan.
- Menetapkan akun IAM individu untuk setiap orang yang mengelola sumber daya Amazon Relational Database Service (Amazon RDS). Jangan pernah menggunakan pengguna root AWS akun untuk mengelola sumber daya Neptunus. Buat pengguna IAM untuk semua orang, termasuk Anda sendiri.
- Berikan setiap pengguna set izin minimum yang diperlukan untuk melakukan tugas mereka.
- Gunakan grup IAM untuk mengelola izin secara efektif bagi beberapa pengguna.
- Putar kredensial IAM Anda secara rutin.

Untuk informasi lebih lanjut tentang menggunakan IAM untuk mengakses sumber daya Neptune, lihat Mengamankan basis data Amazon Neptunus Anda. Untuk informasi umum tentang bekerja dengan IAM, lihat AWS Identity and Access Management dan Praktik Terbaik IAM dalam Panduan Pengguna IAM.

Menghindari kelas instans yang berbeda dalam sebuah klaster

Ketika klaster DB Anda berisi instans dalam kelas yang berbeda, masalah dapat terjadi dari waktu ke waktu. Masalah yang paling umum adalah bahwa instans pembaca kecil bisa masuk ke siklus restart berulang karena perlambatan replikasi. Jika simpul pembaca memiliki konfigurasi kelas instans DB yang lebih lemah dari instans DB penulis, volume perubahan bisa terlalu besar bagi pembaca untuk mengimbanginya.

> **Important**
>
> Untuk menghindari restart berulang yang disebabkan oleh perlambatan replikasi, konfigurasikan klaster DB Anda sehingga semua instans memiliki kelas instans yang sama (ukuran).

Anda dapat melihat jeda antara instance penulis (primer) dan pembaca di cluster DB Anda menggunakan `ClusterReplicaLag` metrik di Amazon CloudWatch. Metrik `VolumeWriteIOPs` juga memungkinkan Anda mendeteksi aktivitas tulis di klaster Anda yang dapat menyebabkan perlambatan replikasi.

Hindari restart berulang selama pemuatan massal

Jika Anda mengalami siklus restart replika baca berulang karena kelambatan replikasi selama pemuatan massal, replika Anda kemungkinan tidak dapat mengikuti penulis di cluster DB Anda.

Baik skala pembaca menjadi lebih besar dari penulis, atau hapus sementara selama pemuatan massal dan kemudian buat ulang setelah selesai.

Aktifkan Indeks OSGP jika Anda memiliki sejumlah besar predikat

Jika model data Anda berisi sejumlah besar predikat yang berbeda (lebih dari seribu dalam banyak kasus), Anda mungkin mengalami penurunan kinerja dan biaya operasional yang lebih tinggi.

Jika ini masalahnya, Anda dapat meningkatkan kinerja dengan mengaktifkan indeks OSGP. Lihat Indeks OSGP.

Menghindari transaksi yang berjalan lama jika memungkinkan

Transaksi berjalan lama pada instans pembaca atau instans penulis dengan penulisan bersamaan, dapat menyebabkan masalah tak terduga dari jenis berikut:

Transaksi berjalan lama pada instans pembaca atau instans penulis dengan penulisan bersamaan dapat mengakibatkan akumulasi besar versi yang berbeda dari data. Hal ini dapat memunculkan latensi yang lebih tinggi untuk kueri baca yang menyaring sebagian besar hasil mereka.

Dalam beberapa kasus, versi akumulasi selama berjam-jam dapat menyebabkan penulisan baru mengalami penyempitan.

Sebuah transaksi baca-tulis yang berjalan lama dengan banyak penulisan juga dapat menyebabkan masalah jika instans dimulai ulang. Jika instans dimulai ulang dari peristiwa pemeliharaan atau kerusakan, semua penulisan yang tidak diterapkan dikembalikan. Operasi pembatalan biasanya berjalan di latar belakang dan tidak memblokir instans dari aktif kembali, tetapi setiap penulisan baru yang berkonflik dengan operasi yang dikembalikan akan gagal.

Sebagai contoh, jika kueri yang sama dicoba lagi setelah sambungan terputus dalam proses sebelumnya mungkin gagal ketika instans dimulai ulang.

Waktu yang dibutuhkan untuk operasi pembatalan sebanding dengan ukuran perubahan yang terlibat.

Praktik terbaik untuk menggunakan metrik Neptunus

Untuk mengidentifikasi masalah performa yang disebabkan sumber daya yang tidak mencukupi dan kemacetan umum lainnya, Anda dapat memantau metrik yang tersedia untuk klaster DB Neptune.

Pantau metrik performa secara rutin untuk mengumpulkan data tentang nilai rata-rata, maksimum, dan minimum untuk berbagai rentang waktu. Hal ini membantu mengidentifikasi saat performa menurun. Dengan menggunakan data ini, Anda dapat mengatur CloudWatch alarm Amazon untuk ambang metrik tertentu sehingga Anda diberi tahu jika tercapai.

Ketika Anda menyiapkan klaster DB baru dan menjalankannya dengan beban kerja tipikal, cobalah menangkap nilai rata-rata, maksimum, dan minimum dari semua metrik kinerja pada sejumlah interval yang berbeda (misalnya, satu jam, 24 jam, satu minggu, dua minggu) untuk mendapatkan gambaran tentang apa yang normal. Ini memberi Anda gambaran tentang apa yang normal. Hal ini membantu mendapatkan perbandingan untuk jam sibuk dan tidak sibuk. Kemudian, Anda dapat menggunakan informasi ini untuk mengidentifikasi saat performa turun di bawah tingkat standar, dan dapat menyesuaikan alarm.

Lihat Memantau Neptunus Menggunakan Amazon CloudWatch untuk informasi tentang cara melihat metrik Neptune.

Berikut ini adalah metrik paling penting untuk memulai:

- BufferCacheHitRatio— Persentase permintaan yang dilayani oleh cache buffer. Cache terlewat menambahkan latensi signifikan untuk eksekusi kueri. Jika rasio hit cache di bawah 99,9% dan laten menjadi masalah untuk aplikasi Anda, pertimbangkan untuk meningkatkan tipe instans untuk meng-cache lebih banyak data dalam memori.
- Pemanfaatan CPU — Persentase kegiatan pemrosesan komputer yang digunakan. Nilai tinggi untuk konsumsi CPU mungkin sesuai, tergantung tujuan performa kueri Anda.
- Memori yang bisa dibebaskan — Berapa banyak RAM yang tersedia di instans DB, dalam megabyte. Neptune memiliki manajer memori sendiri, jadi metrik ini mungkin lebih rendah dari yang Anda harapkan. Pertanda baik bahwa Anda harus mempertimbangkan untuk memutakhirkan kelas instans Anda ke kelas dengan lebih banyak RAM adalah jika kueri sering kali memberikan pengecualian. out-of-memory

Garis merah dalam metrik tab Pemantauan ditandai pada 75% untuk Metrik CPU dan Memori. Jika konsumsi memori instans sering melewati baris tersebut, periksa beban kerja Anda dan pertimbangkan untuk meningkatkan performa kueri.

Praktik terbaik untuk menyetel kueri Neptunus

Salah satu cara terbaik untuk meningkatkan performa Neptune adalah dengan menyetel kueri yang paling sering digunakan dan paling intensif sumber daya untuk membuatnya lebih murah untuk dijalankan.

Untuk informasi tentang cara menyetel kueri Gremlin, lihat Petunjuk kueri Gremlin dan Menyetel kueri Gremlin. Untuk informasi tentang cara menyetel kueri SPARQL, lihat Petunjuk kueri SPARQL.

Load balancing di seluruh replika baca

Perutean round-robin reader endpoint bekerja dengan mengubah host yang ditunjuk entri DNS. Klien harus membuat koneksi baru dan menyelesaikan catatan DNS untuk mendapatkan koneksi ke replika baca baru, karena WebSocket koneksi sering disimpan hidup untuk waktu yang lama.

Untuk mendapatkan replika baca yang berbeda untuk permintaan berturut-turut, pastikan klien Anda menyelesaikan entri DNS setiap kali tersambung. Ini mungkin memerlukan penutupan koneksi dan menyambung kembali ke reader endpoint.

Anda juga dapat menyeimbangkan beban permintaan di seluruh replika baca dengan menyambung ke titik akhir instans secara eksplisit.

Memuat lebih cepat menggunakan instance sementara yang lebih besar

Performa beban Anda meningkat dengan ukuran instans yang lebih besar. Jika Anda tidak menggunakan tipe instans besar, tetapi ingin meningkatkan kecepatan beban, Anda dapat menggunakan instans yang lebih besar untuk dimuat, lalu menghapusnya.

> **Note**
>
> Prosedur berikut berlaku untuk klaster baru. Jika Anda sudah memiliki klaster, Anda dapat menambahkan instans baru yang lebih besar lalu menaikkannya ke instans DB primer.

Untuk memuat data menggunakan ukuran instans yang lebih besar

1. Buat sebuah klaster dengan satu instans `r5.12xlarge`. Instans ini adalah instans DB primer.
2. Buat satu replika baca atau lebih dengan ukuran yang sama (`r5.12xlarge`).

Anda dapat membuat replika baca dalam ukuran yang lebih kecil, tetapi jika replika tersebut tidak cukup besar untuk mengimbangi penulisan yang dibuat oleh instans primer, replika mungkin sering restart. Waktu henti yang dihasilkan mengurangi kinerja secara dramatis.

3. Dalam perintah loader massal, sertakan `“parallelism” : “OVERSUBSCRIBE”` untuk memberi tahu Neptune agar menggunakan semua sumber daya CPU yang tersedia untuk memuat (lihat Parameter Permintaan Loader Neptune). Operasi beban kemudian akan dilanjutkan secepat I/O izin, yang umumnya membutuhkan 60-70% sumber daya CPU.
4. Muat data Anda menggunakan loader Neptune. Tugas pemuatan berjalan pada instans DB primer.
5. Setelah data selesai dimuat, pastikan untuk menskalakan semua instans di klaster ke tipe instans yang sama untuk menghindari biaya tambahan dan masalah restart berulang (lihat Menghindari ukuran instans yang berbeda).

Mengubah ukuran instans penulis Anda dengan melakukan failover pada replika-baca

Cara terbaik untuk mengubah ukuran instans dalam cluster DB Anda, termasuk instans penulis, adalah membuat atau memodifikasi instans replika-baca agar memiliki ukuran yang Anda inginkan, lalu sengaja melakukan failover pada replika-baca tersebut. Waktu henti yang dilihat oleh aplikasi Anda hanyalah waktu yang diperlukan untuk mengubah alamat IP penulis, yang seharusnya sekitar 3 sampai 5 detik.

API manajemen Neptune yang Anda gunakan untuk melakukan failover pada instans penulis saat ini ke instans replika-baca secara sengaja adalah Kegagalan DBCluster. Jika Anda menggunakan klien Gremlin Java, Anda mungkin perlu membuat Objek klien baru setelah failover untuk mengambil alamat IP baru, seperti yang disebutkan di sini.

Pastikan untuk mengubah semua instans Anda ke ukuran yang sama agar Anda terhindar dari siklus restart berulang, seperti yang disebutkan di bawah ini.

Coba lagi unggah setelah kesalahan terputus tugas prefetch data

Saat Anda memuat data ke Neptune menggunakan loader massal, status `LOAD_FAILED` kadang-kadang dapat menghasilkan, dengan pesan `PARSING_ERROR` dan `Data prefetch task interrupted` yang dilaporkan dalam menanggapi permintaan untuk informasi yang mendetail, seperti ini:

```
"errorLogs" : [
  {
    "errorCode" : "PARSING_ERROR",
    "errorMessage" : "Data prefetch task interrupted: Data prefetch task for 11467
 failed",
    "fileName" : "s3://amzn-s3-demo-bucket/some-source-file",
    "recordNum" : 0
  }
]
```

Jika Anda mengalami kesalahan ini, coba lagi permintaan unggah massal lagi.

Kesalahan terjadi ketika ada gangguan sementara yang biasanya tidak disebabkan oleh permintaan atau data Anda, dan biasanya dapat diselesaikan dengan menjalankan permintaan unggah massal lagi.

Jika Anda menggunakan pengaturan default, yaitu `"mode":"AUTO"`, dan `"failOnError":"TRUE"`, loader melewatkan file yang sudah berhasil dimuat dan melanjutkan pemuatan file yang belum dimuat saat terjadi gangguan.

Praktik Terbaik Umum untuk Menggunakan Gremlin dengan Neptune

Ikuti rekomendasi ini saat menggunakan bahasa traversal grafik Gremlin dengan Neptune. Untuk informasi tentang menggunakan Gremlin di Neptune, lihat the section called “Gremlin”.

Important

Perubahan dibuat di TinkerPop versi 3.4.11 yang meningkatkan kebenaran bagaimana kueri diproses, tetapi untuk saat ini terkadang dapat berdampak serius pada kinerja kueri. Misalnya, kueri semacam ini dapat berjalan jauh lebih lambat:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  out()
```

Simpul setelah langkah batas sekarang diambil dengan cara yang tidak optimal karena perubahan 3.4.11. TinkerPop Untuk menghindari hal ini, Anda dapat memodifikasi kueri dengan menambahkan langkah penghalang () kapan saja setelah`order().by()`. Contoh:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

TinkerPop 3.4.11 diaktifkan di mesin Neptunus versi 1.0.5.0.

Topik

- Konfigurasi Detak Jantung untuk Neptunus Tanpa Server
- Struktur meningkatkan kueri untuk memanfaatkan mesin DFE
- Uji kode Gremlin dalam konteks Anda akan menyebarkannya
- Membuat Penulisan Gremlin Multithreaded yang Efisien
- Pemangkasan Catatan dengan Properti Waktu Pembuatan
- Menggunakan Metode datetime( ) untuk Waktu Data Groovy
- Menggunakan Tanggal dan Waktu Asli untuk Data Waktu GLV

Konfigurasi Detak Jantung untuk Neptunus Tanpa Server

Saat menggunakan WebSocket klien Gremlin dengan Neptunus Tanpa Server, Anda perlu mengonfigurasi interval ping klien dengan tepat untuk mempertahankan koneksi yang stabil selama peristiwa penskalaan. Klien Gremlin menggunakan WebSocket koneksi dan mengirimkan ping berkala untuk memverifikasi koneksi aktif. Klien mengharapkan respons dari server dalam jangka waktu interval ping. Jika server tidak merespons, klien secara otomatis menutup koneksi.

Untuk instance yang disediakan Neptunus, sebaiknya atur interval ping menjadi 5 detik. Untuk cluster Neptunus Tanpa Server, sebaiknya atur interval ping setidaknya 20 detik untuk mengakomodasi potensi penundaan selama operasi penskalaan. Parameter ini mengontrol berapa lama klien menunggu antara menulis ke server sebelum mengirim ping untuk memverifikasi koneksi masih aktif.

Konfigurasi parameter ini bervariasi tergantung pada implementasi klien:

Konfigurasi Klien Java

Untuk klien Java TinkerPop Gremlin, konfigurasikan parameter: `keepAliveInterval`

```
Cluster.Builder builder = Cluster.build()
    .addContactPoint(endpoint)
    .keepAliveInterval(20000); // Configure ping interval in milliseconds
```

Untuk detail lebih lanjut tentang konfigurasi driver Java, lihat TinkerPop dokumentasi Java.

Konfigurasi Go Client

Untuk klien Gremlin Go, konfigurasikan parameter: `KeepAliveInterval`

```
rc, err := driver.NewDriverRemoteConnection(endpoint,
    func(settings *driver.DriverRemoteConnectionSettings) {
        settings.TraversalSource = "g"
        settings.AuthInfo = auth
        settings.KeepAliveInterval = 20 * time.Second // Configure ping interval
        ...
    })
```

Untuk detail selengkapnya tentang konfigurasi driver Go, lihat TinkerPop dokumentasi Go.

JavaScriptKonfigurasi Klien /Node.js

Untuk klien Gremlin JavaScript /Node.js, konfigurasikan parameter: `pingInterval`

```
const gremlin = require('gremlin');
const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;

const connection = new DriverRemoteConnection(endpoint, {
    traversalSource: 'g',
    pingInterval: 20000  // Configure ping interval in milliseconds
});
```

Untuk detail lebih lanjut tentang konfigurasi JavaScript driver, lihat JavaScript TinkerPop dokumentasi.

Konfigurasi Klien Python

Untuk klien Python Gremlin, interval ping biasanya dikelola pada lapisan transport. Konsultasikan dokumentasi implementasi transport khusus untuk opsi konfigurasi:

```
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection

g = traversal().with_remote(
    DriverRemoteConnection('wss://your-neptune-endpoint:your-neptune-port/gremlin','g',
        transport_factory=lambda: AiohttpTransport(read_timeout=60,
                                                   write_timeout=20,
                                                   heartbeat=20, // Configure
 heartbeat
                                                   call_from_event_loop=True,
                                                   max_content_length=100*1024*1024,

 ssl_options=ssl.create_default_context(Purpose.CLIENT_AUTH))))
```

Untuk detail lebih lanjut tentang konfigurasi driver Python, lihat dokumentasi Python. TinkerPop

Konfigurasi ini memastikan klien Anda menjaga stabilitas koneksi selama peristiwa penskalaan Neptunus Tanpa Server, mencegah penutupan koneksi yang tidak perlu dan meningkatkan keandalan aplikasi.

Struktur meningkatkan kueri untuk memanfaatkan mesin DFE

Membuat peningkatan yang efisien dengan `mergeV()` Gremlin dan langkah-langkah `mergeE()`menjelaskan cara menyusun kueri upsert untuk menggunakan mesin DFE seefektif mungkin.

Uji kode Gremlin dalam konteks Anda akan menyebarkannya

Di Gremlin, ada beberapa cara bagi klien untuk mengirimkan kueri ke server: menggunakan WebSocket, atau Bytecode GLV, atau melalui konsol Gremlin menggunakan skrip berbasis string.

Penting untuk mengenali bahwa eksekusi kueri Gremlin dapat berbeda tergantung cara Anda mengirimkan kueri. Sebuah kueri yang mengembalikan hasil kosong mungkin diperlakukan sebagai telah berhasil jika dikirimkan dalam mode Bytecode, tetapi diperlakukan sebagai telah gagal jika disampaikan dalam mode script. Misalnya, jika Anda memasukkan `next()` dalam kueri mode skrip, dikirim ke server, tetapi menggunakan ByteCode klien biasanya memproses itu `next()` sendiri. `next()` Dalam kasus pertama, kueri gagal jika tidak ada hasil yang ditemukan, tetapi dalam kasus kedua, kueri berhasil meskupun set hasilnya kosong atau tidak.

Jika Anda mengembangkan dan menguji kode Anda dalam satu konteks (misalnya, konsol Gremlin yang umumnya mengajukan kueri dalam bentuk teks), tetapi kemudian men-deploy kode Anda dalam konteks yang berbeda (misalnya melalui driver Java menggunakan Bytecode), Anda dapat mengalami masalah saat kode Anda berperilaku berbeda dalam produksi alih-alih hal itu terjadi di lingkungan pengembangan Anda.

> **Important**
>
> Pastikan untuk menguji kode Gremlin dalam konteks GLV yang akan ia gunakan, untuk menghindari hasil yang tidak diharapkan.

Membuat Penulisan Gremlin Multithreaded yang Efisien

Ada beberapa pedoman untuk pemuatan multithreaded data ke Neptune menggunakan Gremlin.

Jika memungkinkan, berikan setiap thread satu set simpul atau edge untuk penyisipan atau modifikasi yang tidak bertabrakan. Sebagai contoh, thread 1 menangani rentang ID 1–50.000, thread 2 menangani rentang ID 50,001–100.000, dan seterusnya. Hal ini mengurangi kemungkinan memperoleh `ConcurrentModificationException`. Agar aman, letakkan blok `try/catch` di sekitar semua penulisan. Jika ada yang gagal, Anda bisa mencobanya lagi setelah beberapa saat.

Penulisan dalam batch dalam ukuran batch antara 50 dan 100 (vertex atau edge) umumnya bekerja dengan baik. Jika Anda memiliki banyak properti yang ditambahkan untuk setiap vertex, angka yang mendekati 50 dari 100 mungkin menjadi pilihan yang lebih baik. Beberapa eksperimen sangat berharga. Jadi untuk penulisan dalam batch, Anda dapat menggunakan sesuatu seperti ini:

```
g.addV('test').property(id,'1').as('a').
  addV('test').property(id,'2').
  addE('friend').to('a').
```

Hal ini kemudian diulang dalam setiap operasi batch.

Menggunakan batch secara signifikan lebih efisien alih-alih menambahkan satu vertex atau edge per perjalanan memutar Gremlin ke server.

Jika menggunakan klien Gremlin Language Variant (GLV), Anda dapat membuat batch pemrograman dengan terlebih dahulu membuat traversal. Kemudian tambahkan ke dalamnya, dan akhirnya, ulangi di atasnya; misalnya:

```
t.addV('test').property(id,'1').as('a')
t.addV('test').property(id,'2')
t.addE('friend').to('a')
t.iterate()
```

Sebaiknya gunakan klien Gremlin Language Variant jika memungkinkan. Namun Anda dapat melakukan sesuatu yang mirip dengan klien yang mengajukan kueri sebagai string teks dengan menggabungkan string untuk membangun batch.

Jika Anda menggunakan salah satu pustaka Gremlin klien alih-alih HTTP dasar untuk kueri, semua thread harus berbagi klien, cluster, atau kolam koneksi yang sama. Anda mungkin perlu menyetel pengaturan untuk mendapatkan kemungkinan throughput terbaik—pengaturan seperti ukuran kolam koneksi dan jumlah thread pekerja yang digunakan klien Gremlin.

Pemangkasan Catatan dengan Properti Waktu Pembuatan

Anda dapat memangkas catatan lawas dengan menyimpan waktu pembuatan sebagai properti pada vertex dan meletakkannya secara berkala.

Jika Anda perlu menyimpan data untuk masa hidup tertentu lalu menghapusnya dari grafik (waktu untuk tayang vertex), Anda dapat menyimpan properti timestamp pada pembuatan vertex. Anda kemudian dapat secara berkala mengeluarkan kueri `drop()` untuk semua vertex yang dibuat sebelum waktu tertentu; sebagai contoh:

```
g.V().has("timestamp", lt(datetime('2018-10-11')))
```

Menggunakan Metode **`datetime( )`** untuk Waktu Data Groovy

Neptune menyediakan metode `datetime` untuk menentukan tanggal dan waktu bagi kueri yang dikirim dalam varian Groovy Gremlin. Ini termasuk Konsol Gremlin, string teks menggunakan API REST HTTP, dan serialisasi lain yang menggunakan Groovy.

> **Important**
>
> Ini hanya berlaku untuk metode di mana Anda mengirim kueri Gremlin sebagai string teks. Jika Anda menggunakan Gremlin Language Variant, Anda harus menggunakan kelas tanggal dan fungsi asli untuk bahasanya. Untuk informasi lebih lanjut, lihat bagian selanjutnya, the section called "Tanggal dan Waktu Asli".

Dimulai dengan TinkerPop `3.5.2` (diperkenalkan dalam rilis mesin Neptunus 1.1.1.0)`datetime`, merupakan bagian integral dari. TinkerPop

Anda dapat menggunakan metode `datetime` untuk menyimpan dan membandingkan tanggal:

```
g.V('3').property('date',datetime('2001-02-08'))
```

```
g.V().has('date',gt(datetime('2000-01-01')))
```

Menggunakan Tanggal dan Waktu Asli untuk Data Waktu GLV

Jika Anda menggunakan Gremlin Language Variant (GLV), Anda harus menggunakan kelas tanggal dan waktu serta fungsi asli yang disediakan oleh bahasa pemrograman untuk data waktu Gremlin.

TinkerPop Perpustakaan resmi semuanya adalah perpustakaan Varian Bahasa Gremlin.

- Go
- Java
- Javascript
- .NET
- Python

Important

Halaman ini hanya berlaku untuk pustaka Gremlin Language Variant (GLV). Jika Anda menggunakan metode di mana Anda mengirim kueri Gremlin sebagai string teks, Anda harus menggunakan fungsi datetime () Gremlin. Ini termasuk Konsol Gremlin, string teks menggunakan HTTP REST API atau langsung mengirimkan string Gremlin melalui driver.

Go

Berikut ini adalah contoh sebagian di Go yang membuat properti tunggal bernama 'date' untuk simpul dengan ID '3'. Ini menetapkan nilai menjadi tanggal yang dihasilkan menggunakan fungsi Go time.now ().

```
import ( "time" )

g.V('3').property('date', time.Now()).next();
```

Untuk contoh lengkap untuk menghubungkan ke Neptunus menggunakan Go, lihat Menggunakan klien Go untuk terhubung ke instans DB Neptunus.

Java

Berikut ini adalah contoh parsial dalam Java yang menciptakan satu properti bernama '`date`' untuk vertex dengan ID '3'. Ini menetapkan nilai menjadi tanggal yang dihasilkan menggunakan konstruktor `Date()` Java.

```
import java.util.date

g.V('3').property('date', new Date()).next();
```

Untuk contoh lengkap untuk menghubungkan ke Neptunus menggunakan Java, lihat. Menggunakan klien Java untuk terhubung ke instance DB Neptunus

Node.js (JavaScript)

Berikut ini adalah contoh sebagian dalam JavaScript yang menciptakan properti tunggal bernama '`date`' untuk simpul dengan ID '3'. Ini menetapkan nilai menjadi tanggal yang dihasilkan menggunakan konstruktor `Date()` Node.js.

```
g.V('3').property('date', new Date()).next()
```

Untuk contoh lengkap untuk menghubungkan ke Neptunus menggunakan Node.js, lihat. Gunakan Node.js untuk terhubung ke instans DB Neptune

.NET (C#)

Berikut ini adalah contoh parsial dalam C# yang menciptakan satu properti bernama '`date`' untuk vertex dengan ID '3'. Ini menetapkan nilai menjadi tanggal yang dihasilkan menggunakan properti *`DateTime.UtcNow`* .Net.

```
Using System;

g.V('3').property('date', DateTime.UtcNow).next()
```

Untuk contoh lengkap untuk menghubungkan ke Neptunus menggunakan C #, lihat. Gunakan .NET untuk terhubung ke instans DB Neptune

Python

Berikut ini adalah contoh parsial dalam Python yang menciptakan satu properti bernama '`date`' untuk vertex dengan ID '3'. Ini menetapkan nilai menjadi tanggal yang dihasilkan menggunakan metode `datetime.now()` Python.

```
import datetime

g.V('3').property('date',datetime.datetime.now()).next()
```

Untuk contoh lengkap untuk menghubungkan ke Neptunus menggunakan Python, lihat. Menggunakan Python untuk terhubung ke instans DB Neptune

Praktik terbaik menggunakan klien Gremlin Java dengan Neptunus

Ikuti rekomendasi ini saat menggunakan klien Gremlin Java dengan Neptunus. Praktik terbaik ini membantu Anda mengoptimalkan kinerja, mengelola koneksi secara efektif, dan menghindari jebakan umum saat bekerja dengan driver Java.

Untuk informasi tentang mengonfigurasi interval detak jantung untuk Neptunus Tanpa Server, lihat. Konfigurasi Detak Jantung untuk Neptunus Tanpa Server

Topik

- Gunakan kembali objek klien di beberapa utas
- Buat objek klien Gremlin Java terpisah untuk titik akhir baca dan tulis
- Tambahkan beberapa titik akhir replika baca ke kumpulan koneksi Gremlin Java
- Tutup klien untuk menghindari batas koneksi
- Buat koneksi baru setelah failover
- Setel maxInProcessPerConnection dan maxSimultaneousUsagePerConnection ke nilai yang sama
- Kirim kueri ke server sebagai bytecode bukan sebagai string
- Selalu gunakan sepenuhnya ResultSet atau Iterator yang dikembalikan oleh kueri
- Massal menambahkan simpul dan tepi dalam batch
- Menonaktifkan Caching DNS di Mesin Virtual Java
- Secara opsional, atur batas waktu pada tingkat per kueri

- Pecahkan masalah java.util.concurrent.TimeoutException

Gunakan kembali objek klien di beberapa utas

Menggunakan objek klien yang sama (atau `GraphTraversalSource`) di beberapa thread. Yaitu, membuat instans bersama dari kelas `org.apache.tinkerpop.gremlin.driver.Client` dalam aplikasi Anda alih-alih melakukannya di setiap thread. Objek `Client` aman untuk thread, dan overhead yang menginisialisasinya cukup besar.

Ini juga berlaku untuk `GraphTraversalSource`, yang menciptakan objek `Client` secara internal. Misalnya, kode berikut menyebabkan objek `Client` baru akan dipakai:

```
import static
 org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource.traversal;

  /////

GraphTraversalSource traversal = traversal()
                                    .withRemote(DriverRemoteConnection.using(cluster));
```

Buat objek klien Gremlin Java terpisah untuk titik akhir baca dan tulis

Anda dapat meningkatkan performa dengan hanya melakukan penulisan pada titik akhir tulis dan baca dari satu titik akhir baca-saja atau lebih.

```
Client readerClient = Cluster.build("https://reader-endpoint")
          ...
          .connect()

Client writerClient = Cluster.build("https://writer-endpoint")
          ...
          .connect()
```

Tambahkan beberapa titik akhir replika baca ke kumpulan koneksi Gremlin Java

Saat membuat objek `Cluster` Java Gremlin, Anda dapat menggunakan metode `.addContactPoint()` untuk menambahkan beberapa instans replika baca ke titik kontak kolam koneksi.

```
Cluster.Builder readerBuilder = Cluster.build()
          .port(8182)
          .minConnectionPoolSize(…)
          .maxConnectionPoolSize(…)
          ……
          .addContactPoint("reader-endpoint-1")
          .addContactPoint("reader-endpoint-2")
```

Tutup klien untuk menghindari batas koneksi

Penting untuk menutup klien ketika Anda selesai dengan itu untuk memastikan bahwa WebSocket koneksi ditutup oleh server dan semua sumber daya yang terkait dengan koneksi dilepaskan. Hal ini terjadi secara otomatis jika Anda menutup klaster menggunakan `Cluster.close( )`, karena `client.close( )` kemudian dipanggil secara internal.

Jika klien tidak ditutup dengan benar, Neptunus menghentikan semua koneksi WebSocket idle setelah 20 hingga 25 menit. Namun, jika Anda tidak secara eksplisit menutup WebSocket koneksi ketika Anda selesai dengan mereka dan jumlah koneksi langsung mencapai batas koneksi WebSocket bersamaan, koneksi tambahan kemudian ditolak dengan kode kesalahan HTTP429. Pada saat itu, Anda harus me-restart instans Neptune untuk menutup koneksi.

Saran untuk memanggil `cluster.close()` tidak berlaku untuk fungsi AWS Lambda Java. Lihat Mengelola WebSocket koneksi Gremlin dalam fungsi AWS Lambda untuk detail.

Buat koneksi baru setelah failover

Dalam kasus failover, Driver Gremlin mungkin terus menyambung ke penulis yang lama karena nama DNS klaster diselesaikan ke alamat IP. Jika hal ini terjadi, Anda dapat membuat objek `Client` baru setelah failover.

Setel **`maxInProcessPerConnection`** dan **`maxSimultaneousUsagePerConnection`** ke nilai yang sama

Baik parameter `maxInProcessPerConnection` dan `maxSimultaneousUsagePerConnection` parameter terkait dengan jumlah maksimum kueri simultan yang dapat Anda kirimkan pada satu WebSocket koneksi. Secara internal, parameter ini terkait bersama dan memodifikasi salah satunya tanpa yang lain dapat menyebabkan klien menerima timeout ketika mencoba mengambil koneksi dari kolam koneksi klien.

Sebaiknya pertahankan nilai minimum default dalam proses dan penggunaan simultan, dan pengaturan `maxInProcessPerConnection` dan `maxSimultaneousUsagePerConnection` dengan nilai yang sama.

Nilai untuk mengatur parameter ini adalah fungsi dari kompleksitas kueri dan model data. Kasus penggunaan saat kueri yang mengembalikan banyak data akan membutuhkan lebih banyak bandwidth koneksi per kueri dan karenanya, harus memiliki nilai yang lebih rendah untuk parameternya, dan nilai yang lebih tinggi untuk `maxConnectionPoolSize`.

Sebaliknya, dalam kasus saat kueri mengembalikan jumlah yang lebih kecil dari data, `maxInProcessPerConnection` dan `maxSimultaneousUsagePerConnection` harus diatur ke nilai yang lebih tinggi dari `maxConnectionPoolSize`.

Kirim kueri ke server sebagai bytecode bukan sebagai string

Ada keuntungan menggunakan bytecode alih-alih string saat mengirimkan kueri:

- Menangkap sintaks kueri yang tidak valid di awal: Menggunakan varian bytecode memungkinkan Anda mendeteksi sintaks kueri yang tidak valid pada tahap kompilasi. Jika menggunakan variasi berbasis string, Anda tidak akan menemukan sintaks yang tidak valid sampai kueri dikirimkan ke server dan kesalahan dikembalikan.
- Hindari penalti kinerja berbasis string: Setiap pengiriman kueri berbasis string, apakah Anda menggunakan WebSockets atau HTTP, menghasilkan simpul terpisah, yang menyiratkan bahwa objek Vertex terdiri dari ID, Label, dan semua properti yang terkait dengan Vertex (lihat Properti Elemen).

  Hal ini dapat menyebabkan komputasi yang tidak perlu pada server dalam kasus saat properti tidak diperlukan. Misalnya, jika pelanggan tertarik untuk mendapatkan vertex dengan ID “hakuna#1” menggunakan kueri, `g.V("hakuna#1")`. Jika kueri dikirim sebagai pengiriman berbasis string, server akan menghabiskan waktu dalam mengambil ID, label, dan semua properti untuk vertex ini. Jika kueri dikirim sebagai pengiriman bytecode, server hanya menghabiskan waktu untuk mengambil ID dan label dari vertex.

Dengan kata lain, alih-alih mengirimkan kueri seperti ini:

```
final Cluster cluster = Cluster.build("localhost")
                               .port(8182)
                               .maxInProcessPerConnection(32)
                               .maxSimultaneousUsagePerConnection(32)
```

```
                                .serializer(Serializers.GRAPHBINARY_V1D0)
                                .create();

 try {
     final Client client = cluster.connect();
     List<Result> results =
client.submit("g.V().has('name','pumba').out('friendOf').id()").all().get();
     System.out.println(verticesWithNamePumba);
 } finally {
     cluster.close();
 }
```

Sebaliknya, kirimkan kueri menggunakan bytecode, seperti ini:

```
 final Cluster cluster = Cluster.build("localhost")
                                .port(8182)
                                .maxInProcessPerConnection(32)
                                .maxSimultaneousUsagePerConnection(32)
                                .serializer(Serializers.GRAPHBINARY_V1D0)
                                .create();

 try {
     final GraphTraversalSource g =
traversal().withRemote(DriverRemoteConnection.using(cluster));
     List<Object> verticesWithNamePumba = g.V().has("name",
"pumba").out("friendOf").id().toList();
     System.out.println(verticesWithNamePumba);
 } finally {
     cluster.close();
 }
```

Selalu gunakan sepenuhnya ResultSet atau Iterator yang dikembalikan oleh kueri

Objek klien harus selalu benar-benar mengonsumsi `ResultSet` (dalam kasus pengiriman berbasis string), atau iterator yang dikembalikan oleh `GraphTraversal`. Jika hasil kueri tidak benar-benar dikonsumsi, server bergantung pada mereka, menunggu klien untuk menyelesaikan mengkonsumsi mereka.

Jika aplikasi Anda hanya membutuhkan sebagian hasil, Anda dapat menggunakan langkah `limit(X)` dengan kueri Anda untuk membatasi jumlah hasil yang server hasilkan.

Massal menambahkan simpul dan tepi dalam batch

Setiap kueri ke Neptune DB berjalan dalam lingkup transaksi tunggal, kecuali jika Anda menggunakan sesi. Ini berarti bahwa jika Anda perlu memasukkan banyak data menggunakan kueri Gremlin, menggabungkan mereka bersama-sama dalam ukuran batch 50-100 akan meningkatkan kinerja dengan mengurangi jumlah transaksi yang dibuat untuk beban.

Sebagai contoh, menambahkan 5 vertex ke basis data akan terlihat seperti ini:

```
// Create a GraphTraversalSource for the remote connection
final GraphTraversalSource g =
 traversal().withRemote(DriverRemoteConnection.using(cluster));
// Add 5 vertices in a single query
g.addV("Person").property(T.id, "P1")
 .addV("Person").property(T.id, "P2")
 .addV("Person").property(T.id, "P3")
 .addV("Person").property(T.id, "P4")
 .addV("Person").property(T.id, "P5").iterate();
```

Demikian pula, Anda dapat menambahkan tepi secara batch menggunakan. `addE` Gunakan `V()` untuk mereferensikan simpul yang ada sebagai sumber dan target dari setiap tepi:

```
// Add edges in a single batched query
g.V("P1").addE("knows").to(V("P2"))
 .V("P2").addE("knows").to(V("P3"))
 .V("P3").addE("knows").to(V("P4"))
 .V("P4").addE("knows").to(V("P5")).iterate();
```

Anda juga dapat menggabungkan pembuatan simpul dan tepi dalam satu batch. Gunakan `as()` untuk memberi label simpul yang baru dibuat sehingga Anda dapat mereferensikannya saat menambahkan tepi dalam traversal yang sama:

```
// Add vertices and edges together in a single query
g.addV("Person").property(T.id, "P1").as("p1")
 .addV("Person").property(T.id, "P2").as("p2")
 .addV("Person").property(T.id, "P3").as("p3")
 .addE("knows").from("p1").to("p2")
 .addE("knows").from("p2").to("p3").iterate();
```

Menonaktifkan Caching DNS di Mesin Virtual Java

Dalam lingkungan di mana Anda ingin memuat permintaan keseimbangan di beberapa replika baca, Anda perlu menonaktifkan cache DNS di Java Virtual Machine (JVM) dan menyediakan titik akhir pembaca Neptunus saat membuat objek Cluster. Menonaktifkan cache DNS JVM memastikan bahwa DNS diselesaikan lagi untuk setiap koneksi baru sehingga permintaan didistribusikan di semua replika baca. Anda dapat melakukan ini dalam kode inisialisasi aplikasi Anda dengan baris berikut:

```
java.security.Security.setProperty("networkaddress.cache.ttl", "0");
```

Namun, solusi yang lebih lengkap dan kuat untuk load-balancing disediakan oleh kode klien Amazon Gremlin Java pada. GitHub Klien Amazon Java Gremlin menyadari topologi klaster Anda dan cukup mendistribusikan koneksi dan permintaan pada satu set instans di klaster Neptune Anda. Lihat postingan blog ini untuk sampel fungsi Java Lambda yang menggunakan klien tersebut.

Secara opsional, atur batas waktu pada tingkat per kueri

Neptune menyediakan kemampuan untuk mengatur batas waktu untuk kueri Anda menggunakan opsi grup parameter `neptune_query_timeout` (lihat Parameter). Anda juga dapat mengganti batas waktu global, dengan kode seperti ini:

```
 final Cluster cluster = Cluster.build("localhost")
                               .port(8182)
                               .maxInProcessPerConnection(32)
                               .maxSimultaneousUsagePerConnection(32)
                               .serializer(Serializers.GRAPHBINARY_V1D0)
                               .create();

 try {
     final GraphTraversalSource g =
traversal().withRemote(DriverRemoteConnection.using(cluster));
     List<Object> verticesWithNamePumba = g.with(ARGS_EVAL_TIMEOUT,
500L).V().has("name", "pumba").out("friendOf").id().toList();
     System.out.println(verticesWithNamePumba);
 } finally {
     cluster.close();
 }
```

Atau, untuk pengiriman kueri berbasis string, kodenya akan terlihat seperti ini:

```
RequestOptions options = RequestOptions.build().timeout(500).create();
List<Result> result = client.submit("g.V()", options).all().get();
```

> **Note**
>
> Dimungkinkan untuk mengeluarkan biaya tak terduga jika Anda menetapkan nilai batas waktu kueri terlalu tinggi, terutama pada instance tanpa server. Tanpa pengaturan batas waktu yang wajar, kueri Anda dapat terus berjalan lebih lama dari yang Anda harapkan, menimbulkan biaya yang tidak pernah Anda antisipasi. Hal ini terutama berlaku pada instance tanpa server yang dapat meningkatkan skala hingga jenis instance yang besar dan mahal saat menjalankan kueri.
>
> Anda dapat menghindari pengeluaran tak terduga semacam ini dengan menggunakan nilai batas waktu kueri yang mengakomodasi run-time yang Anda harapkan dan hanya menyebabkan jangka waktu habis yang luar biasa.
>
> Mulai dari mesin Neptunus versi 1.3.2.0, Neptunus mendukung parameter neptune_lab_mode baru sebagai parameter. `StrictTimeoutValidation` Ketika parameter ini memiliki nilai`Enabled`, nilai batas waktu per kueri yang ditentukan sebagai opsi permintaan atau petunjuk kueri tidak dapat melebihi nilai yang ditetapkan secara global dalam grup parameter. Dalam kasus seperti itu, Neptunus akan melempar. `InvalidParameterException` Pengaturan ini dapat dikonfirmasi dalam respons pada titik akhir '/status' saat nilainya. `Disabled` Dalam versi mesin`1.3.2.0`, nilai default parameter ini adalah`Disabled`. Mulai dalam versi mesin`1.4.0.0`, `StrictTimeoutValidation` parameternya secara `Enabled` default.
>
> Untuk informasi selengkapnya tentang bagaimana prioritas batas waktu ditentukan ketika beberapa pengaturan batas waktu dikonfigurasi, lihat dokumentasi parameter neptune_query_timeout.

Pecahkan masalah **`java.util.concurrent.TimeoutException`**

Klien Gremlin Java melempar `java.util.concurrent.TimeoutException` ketika permintaan Gremlin habis waktu pada klien itu sendiri sambil menunggu slot di salah satu koneksi menjadi tersedia. WebSocket Durasi batas waktu ini dikendalikan oleh parameter konfigurasi di sisi klien `maxWaitForConnection`.

> **Note**
>
> Karena permintaan yang kehabisan waktu di klien tidak pernah dikirim ke server, mereka tidak tercermin dalam salah satu metrik yang direkam di server, seperti `GremlinRequestsPerSec`.

Batas waktu semacam ini umumnya disebabkan oleh salah satu dari dua cara berikut:

- Server sebenarnya mencapai kapasitas maksimum. Jika ini masalahnya, antrian di server terisi, yang dapat Anda deteksi dengan memantau metrik. MainRequestQueuePendingRequests CloudWatch Jumlah kueri paralel yang dapat ditangani server bergantung pada ukuran instansnya.

  Jika metrik `MainRequestQueuePendingRequests` tidak menampilkan penumpukan permintaan tertunda di server, maka server dapat menangani lebih banyak permintaan dan batas waktu disebabkan oleh throttling sisi klien.
- Pelambatan permintaan klien. Hal ini umumnya dapat diperbaiki dengan mengubah pengaturan konfigurasi klien.

  Jumlah maksimum permintaan paralel yang dapat dikirim klien dapat diperkirakan secara kasar sebagai berikut:

  ```
  maxParallelQueries = maxConnectionPoolSize * Max( maxSimultaneousUsagePerConnection,
   maxInProcessPerConnection )
  ```

  Mengirim lebih dari `maxParallelQueries` ke klien menyebabkan pengecualian `java.util.concurrent.TimeoutException`. Anda biasanya dapat memperbaikinya dengan beberapa cara:

  - Tingkatkan durasi batas waktu koneksi. Jika latensi tidak penting untuk aplikasi Anda, tingkatkan pengaturan `maxWaitForConnection` klien. Klien kemudian menunggu lebih lama sebelum waktu habis, yang pada gilirannya dapat meningkatkan latensi.
  - Tingkatkan permintaan maksimum per koneksi. Ini memungkinkan lebih banyak permintaan dikirim menggunakan WebSocket koneksi yang sama. Lakukan ini dengan meningkatkan pengaturan `maxSimultaneousUsagePerConnection` dan `maxInProcessPerConnection` klien. Pengaturan ini umumnya memiliki nilai yang sama.
  - Tingkatkan jumlah koneksi di kolam koneksi. Lakukan ini dengan meningkatkan pengaturan `maxConnectionPoolSize` klien. Biaya peningkatan konsumsi sumber daya, karena setiap

koneksi menggunakan memori dan deskriptor file sistem operasi, dan memerlukan SSL dan jabat tangan selama inisialisasi. WebSocket

Praktik Terbaik Neptunus Menggunakan OpenCypher dan Bolt

Ikuti praktik terbaik ini saat menggunakan bahasa kueri OpenCypher dan protokol Bolt dengan Neptunus. Untuk informasi tentang menggunakan OpenCypher di Neptunus, lihat. Mengakses Grafik Neptunus dengan OpenCypher

Topik

- Buat koneksi baru setelah failover
- Penanganan koneksi untuk aplikasi berumur panjang
- Penanganan koneksi untuk AWS Lambda
- Lebih suka diarahkan ke tepi dua arah dalam kueri
- Neptunus tidak mendukung beberapa kueri bersamaan dalam suatu transaksi
- Tutup objek pengemudi saat Anda selesai
- Gunakan mode transaksi eksplisit untuk membaca dan menulis
- Coba lagi logika untuk pengecualian
- Mengatur beberapa properti sekaligus menggunakan satu klausa SET
- Gunakan kueri berparameter
- Gunakan peta yang diratakan alih-alih peta bersarang di klausa UNWIND
- Tempatkan node yang lebih ketat di sisi kiri dalam ekspresi Variable-Length Path (VLP)
- Hindari pemeriksaan label node yang berlebihan dengan menggunakan nama hubungan granular
- Tentukan label tepi jika memungkinkan
- Hindari menggunakan klausa WITH jika memungkinkan
- Tempatkan filter restriktif sedini mungkin dalam kueri
- Periksa secara eksplisit apakah properti ada
- Jangan gunakan jalur bernama (kecuali jika diperlukan)
- Hindari KUMPULKAN (DISTINCT ())
- Lebih suka fungsi properti daripada pencarian properti individu saat mengambil semua nilai properti
- Lakukan perhitungan statis di luar kueri
- Masukan batch menggunakan UNWIND alih-alih pernyataan individual

- Lebih suka menggunakan kustom IDs untuk node/hubungan
- Hindari melakukan perhitungan ~id dalam kueri
- Memperbarui/Menggabungkan beberapa node

Buat koneksi baru setelah failover

Dalam kasus failover, driver Bolt dapat terus terhubung ke instance penulis lama daripada yang aktif baru, karena nama DNS diselesaikan ke alamat IP tertentu.

Untuk mencegah hal ini, tutup lalu sambungkan kembali `Driver` objek setelah failover apa pun.

Penanganan koneksi untuk aplikasi berumur panjang

Saat membuat aplikasi yang berumur panjang, seperti yang berjalan di dalam container atau di instans Amazon EC2, buat instance `Driver` objek sekali dan kemudian gunakan kembali objek tersebut selama masa pakai aplikasi. Objek `Driver` aman untuk thread, dan overhead yang menginisialisasinya cukup besar.

Penanganan koneksi untuk AWS Lambda

Driver baut tidak disarankan untuk digunakan dalam AWS Lambda fungsi, karena overhead koneksi dan persyaratan manajemennya. Gunakan titik akhir HTTPS sebagai gantinya.

Lebih suka diarahkan ke tepi dua arah dalam kueri

Saat Neptunus melakukan optimasi kueri, tepi dua arah menyulitkan untuk membuat rencana kueri yang optimal. Rencana sub-optimal mengharuskan mesin melakukan pekerjaan yang tidak perlu dan menghasilkan kinerja yang lebih buruk.

Oleh karena itu, gunakan tepi yang diarahkan daripada yang dua arah bila memungkinkan. Misalnya, gunakan:

```
MATCH p=(:airport {code: 'ANC'})-[:route]->(d) RETURN p)
```

bukannya:

```
MATCH p=(:airport {code: 'ANC'})-[:route]-(d) RETURN p)
```

Sebagian besar model data sebenarnya tidak perlu melintasi tepi di kedua arah, sehingga kueri dapat mencapai peningkatan kinerja yang signifikan dengan beralih menggunakan tepi terarah.

Jika model data Anda memang memerlukan melintasi tepi dua arah, buat node pertama (sisi kiri) dalam `MATCH` pola node dengan penyaringan paling ketat.

Ambil contoh, “Temukan saya semua `routes` ke dan dari `ANC` bandara”. Tulis kueri ini untuk memulai di `ANC` bandara, seperti ini:

```
MATCH p=(src:airport {code: 'ANC'})-[:route]-(d) RETURN p
```

Mesin dapat melakukan jumlah minimal pekerjaan untuk memenuhi query, karena node yang paling terbatas ditempatkan sebagai node pertama (sisi kiri) dalam pola. Mesin kemudian dapat mengoptimalkan kueri.

Ini jauh lebih disukai daripada menyaring `ANC` bandara di akhir pola, seperti ini:

```
MATCH p=(d)-[:route]-(src:airport {code: 'ANC'}) RETURN p
```

Ketika node yang paling terbatas tidak ditempatkan pertama kali dalam pola, mesin harus melakukan pekerjaan tambahan karena tidak dapat mengoptimalkan kueri dan harus melakukan pencarian tambahan untuk sampai pada hasil.

Neptunus tidak mendukung beberapa kueri bersamaan dalam suatu transaksi

Meskipun driver Bolt sendiri memungkinkan kueri bersamaan dalam suatu transaksi, Neptunus tidak mendukung beberapa kueri dalam transaksi yang berjalan secara bersamaan. Sebaliknya, Neptunus mengharuskan beberapa kueri dalam transaksi dijalankan secara berurutan, dan bahwa hasil dari setiap kueri sepenuhnya dikonsumsi sebelum kueri berikutnya dimulai.

Contoh di bawah ini menunjukkan cara menggunakan Bolt untuk menjalankan beberapa kueri secara berurutan dalam suatu transaksi, sehingga hasil masing-masing benar-benar dikonsumsi sebelum yang berikutnya dimulai:

```
final String query = "MATCH (n) RETURN n";

try (Driver driver = getDriver(HOST_BOLT, getDefaultConfig())) {
  try (Session session = driver.session(readSessionConfig)) {
    try (Transaction trx = session.beginTransaction()) {
```

```
      final Result res_1 = trx.run(query);
      Assert.assertEquals(10000, res_1.list().size());
      final Result res_2 = trx.run(query);
      Assert.assertEquals(10000, res_2.list().size());
    }
  }
}
```

Tutup objek pengemudi saat Anda selesai

Pastikan untuk menutup klien ketika Anda selesai dengannya, sehingga koneksi Bolt ditutup oleh server dan semua sumber daya yang terkait dengan koneksi dilepaskan. Ini terjadi secara otomatis jika Anda menutup driver menggunakan`driver.close()`.

Jika driver tidak ditutup dengan benar, Neptunus menghentikan semua koneksi Bolt idle setelah 20 menit, atau setelah 10 hari jika Anda menggunakan otentikasi IAM.

Neptunus mendukung tidak lebih dari 1000 koneksi Bolt bersamaan. Jika Anda tidak secara eksplisit menutup koneksi ketika Anda selesai dengan mereka, dan jumlah koneksi langsung mencapai batas 1000, setiap upaya koneksi baru gagal.

Gunakan mode transaksi eksplisit untuk membaca dan menulis

Saat menggunakan transaksi dengan Neptunus dan driver Bolt, yang terbaik adalah secara eksplisit mengatur mode akses untuk transaksi baca dan tulis ke pengaturan yang tepat.

Transaksi hanya-baca

Untuk transaksi hanya-baca, jika Anda tidak meneruskan konfigurasi mode akses yang sesuai saat membangun sesi, tingkat isolasi default digunakan, yaitu isolasi kueri mutasi. Akibatnya, penting bagi transaksi read-only untuk mengatur mode akses secara eksplisit. `read`

Contoh transaksi baca komit otomatis:

```
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.READ)
  .build();
Session session = driver.session(sessionConfig);
try {
  (Add your application code here)
```

```
} catch (final Exception e) {
  throw e;
} finally {
  driver.close()
}
```

Baca contoh transaksi:

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withDefaultAccessMode(AccessMode.READ)
  .build();
driver.session(sessionConfig).readTransaction(
  new TransactionWork<List<String>>() {
    @Override
    public List<String> execute(org.neo4j.driver.Transaction tx) {
      (Add your application code here)
    }
  }
);
```

Dalam kedua kasus, SNAPSHOTisolasi dicapai dengan menggunakan semantik transaksi read-only Neptunus.

Karena replika baca hanya menerima kueri hanya-baca, kueri apa pun yang dikirimkan ke replika baca berjalan di bawah semantik isolasi. SNAPSHOT

Tidak ada pembacaan kotor atau pembacaan yang tidak dapat diulang untuk transaksi hanya-baca.

Transaksi mutasi

Untuk kueri mutasi, ada tiga mekanisme berbeda untuk membuat transaksi tulis, yang masing-masing diilustrasikan di bawah ini:

Contoh transaksi tulis implisit:

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();
driver.session(sessionConfig).writeTransaction(
```

```
  new TransactionWork<List<String>>() {
    @Override
    public List<String> execute(org.neo4j.driver.Transaction tx) {
      (Add your application code here)
    }
  }
);
```

Contoh transaksi tulis komit otomatis:

```
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.Write)
  .build();
Session session = driver.session(sessionConfig);
try {
  (Add your application code here)
} catch (final Exception e) {
    throw e;
} finally {
    driver.close()
}
```

Contoh transaksi tulis eksplisit:

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();
Transaction beginWriteTransaction = driver.session(sessionConfig).beginTransaction();
  (Add your application code here)
beginWriteTransaction.commit();
driver.close();
```

Tingkat isolasi untuk transaksi tulis

- Pembacaan yang dibuat sebagai bagian dari kueri mutasi dijalankan di bawah isolasi `READ COMMITTED` transaksi.
- Tidak ada bacaan kotor untuk bacaan yang dibuat sebagai bagian dari kueri mutasi.

- Catatan dan rentang catatan dikunci saat membaca dalam kueri mutasi.
- Ketika rentang indeks telah dibaca oleh transaksi mutasi, ada jaminan kuat bahwa rentang ini tidak akan dimodifikasi oleh transaksi bersamaan sampai akhir pembacaan.

Kueri mutasi tidak aman untuk utas.

Untuk konflik, lihatResolusi Konflik Menggunakan Lock-Wait Timeout.

Kueri mutasi tidak dicoba ulang secara otomatis jika terjadi kegagalan.

Coba lagi logika untuk pengecualian

Untuk semua pengecualian yang memungkinkan percobaan ulang, umumnya yang terbaik adalah menggunakan strategi backoff dan coba lagi eksponensial yang menyediakan waktu tunggu yang semakin lama di antara percobaan ulang sehingga dapat menangani masalah sementara seperti kesalahan dengan lebih baik. `ConcurrentModificationException` Berikut ini menunjukkan contoh backoff eksponensial dan pola coba lagi:

```
public static void main() {
  try (Driver driver = getDriver(HOST_BOLT, getDefaultConfig())) {
    retriableOperation(driver, "CREATE (n {prop:'1'})")
        .withRetries(5)
        .withExponentialBackoff(true)
        .maxWaitTimeInMilliSec(500)
        .call();
  }
}

protected RetryableWrapper retriableOperation(final Driver driver, final String query){
  return new RetryableWrapper<Void>() {
    @Override
    public Void submit() {
      log.info("Performing graph Operation in a retry manner......");
      try (Session session = driver.session(writeSessionConfig)) {
        try (Transaction trx =  session.beginTransaction()) {
            trx.run(query).consume();
            trx.commit();
        }
      }
      return null;
    }
```

```
    @Override
    public boolean isRetryable(Exception e) {
      if (isCME(e)) {
        log.debug("Retrying on exception.... {}", e);
        return true;
      }
      return false;
    }

    private boolean isCME(Exception ex) {
      return ex.getMessage().contains("Operation failed due to conflicting concurrent
 operations");
    }
  };
}



/**
 * Wrapper which can retry on certain condition. Client can retry operation using this
 class.
 */
@Log4j2
@Getter
public abstract class RetryableWrapper<T> {

  private long retries = 5;
  private long maxWaitTimeInSec = 1;
  private boolean exponentialBackoff = true;

  /**
   * Override the method with custom implementation, which will be called in retryable
 block.
   */
  public abstract T submit() throws Exception;

  /**
   * Override with custom logic, on which exception to retry with.
   */
  public abstract boolean isRetryable(final Exception e);

  /**
   * Define the number of retries.
   *
```

```
   * @param retries -no of retries.
   */
  public RetryableWrapper<T> withRetries(final long retries) {
    this.retries = retries;
    return this;
  }

  /**
   * Max wait time before making the next call.
   *
   * @param time - max polling interval.
   */
  public RetryableWrapper<T> maxWaitTimeInMilliSec(final long time) {
    this.maxWaitTimeInSec = time;
    return this;
  }

  /**
   * ExponentialBackoff coefficient.
   */
  public RetryableWrapper<T> withExponentialBackoff(final boolean expo) {
    this.exponentialBackoff = expo;
    return this;
  }

  /**
   * Call client method which is wrapped in submit method.
   */
  public T call() throws Exception {
    int count = 0;
    Exception exceptionForMitigationPurpose = null;
    do {
      final long waitTime = exponentialBackoff ? Math.min(getWaitTimeExp(retries),
maxWaitTimeInSec) : 0;
      try {
          return submit();
      } catch (Exception e) {
        exceptionForMitigationPurpose = e;
        if (isRetryable(e) && count < retries) {
          Thread.sleep(waitTime);
          log.debug("Retrying on exception attempt - {} on exception cause - {}",
count, e.getMessage());
        } else if (!isRetryable(e)) {
          log.error(e.getMessage());
```

```
          throw new RuntimeException(e);
        }
      }
    } while (++count < retries);

    throw new IOException(String.format(
          "Retry was unsuccessful.... attempts %d. Hence throwing exception " + "back
 to the caller...", count),
          exceptionForMitigationPurpose);
  }

  /*
   * Returns the next wait interval, in milliseconds, using an exponential backoff
   * algorithm.
   */
  private long getWaitTimeExp(final long retryCount) {
    if (0 == retryCount) {
      return 0;
    }
    return ((long) Math.pow(2, retryCount) * 100L);
  }
}
```

Mengatur beberapa properti sekaligus menggunakan satu klausa SET

Alih-alih menggunakan beberapa klausa SET untuk menyetel properti individual, gunakan peta untuk menyetel beberapa properti untuk entitas sekaligus.

Anda dapat menggunakan:

```
MATCH (n:SomeLabel {`~id`: 'id1'})
SET n += {property1 : 'value1',
property2 : 'value2',
property3 : 'value3'}
```

Alih-alih:

```
MATCH (n:SomeLabel {`~id`: 'id1'})
SET n.property1 = 'value1'
SET n.property2 = 'value2'
SET n.property3 = 'value3'
```

Klausa SET menerima salah satu properti atau peta. Jika memperbarui beberapa properti pada satu entitas, menggunakan klausa SET tunggal dengan peta memungkinkan pembaruan dilakukan dalam satu operasi alih-alih beberapa operasi, yang dapat dijalankan dengan lebih efisien.

Gunakan klausa SET untuk menghapus beberapa properti sekaligus

Saat menggunakan bahasa OpenCypher, REMOVE digunakan untuk menghapus properti dari entitas. Di Neptunus, setiap properti yang dihapus memerlukan operasi terpisah, menambahkan latensi kueri. Sebagai gantinya, Anda dapat menggunakan SET dengan peta untuk mengatur semua nilai propertinull, yang di Neptunus setara dengan menghapus properti. Neptunus akan meningkatkan kinerja ketika beberapa properti pada satu entitas diperlukan untuk dihapus.

Gunakan:

```
WITH {prop1: null, prop2: null, prop3: null} as propertiesToRemove
MATCH (n)
SET n += propertiesToRemove
```

Alih-alih:

```
MATCH (n)
REMOVE n.prop1, n.prop2, n.prop3
```

Gunakan kueri berparameter

Disarankan untuk selalu menggunakan query parameterized saat query menggunakan OpenCypher. Mesin kueri dapat memanfaatkan kueri parameter berulang untuk fitur seperti cache paket kueri, di mana pemanggilan berulang dari struktur berparameter yang sama dengan parameter yang berbeda dapat memanfaatkan paket yang di-cache. Paket kueri yang dihasilkan untuk kueri berparameter di-cache dan digunakan kembali hanya ketika selesai dalam 100ms dan tipe parameternya adalah NUMBER, BOOLEAN atau STRING.

Gunakan:

```
MATCH (n:foo) WHERE id(n) = $id RETURN n
```

Dengan parameter:

```
parameters={"id": "first"}
```

```
parameters={"id": "second"}
parameters={"id": "third"}
```

Alih-alih:

```
MATCH (n:foo) WHERE id(n) = "first" RETURN n
MATCH (n:foo) WHERE id(n) = "second" RETURN n
MATCH (n:foo) WHERE id(n) = "third" RETURN n
```

Gunakan peta yang diratakan alih-alih peta bersarang di klausa UNWIND

Struktur bersarang dalam dapat membatasi kemampuan mesin kueri untuk menghasilkan rencana kueri yang optimal. Untuk mengatasi sebagian masalah ini, pola yang ditentukan berikut akan membuat rencana optimal untuk skenario berikut:

- Skenario 1: BERSANTAILAH dengan daftar literal cypher, yang mencakup NUMBER, STRING dan BOOLEAN.
- Skenario 2: BERSANTAILAH dengan daftar peta yang diratakan, yang hanya mencakup literal cypher (NUMBER, STRING, BOOLEAN) sebagai nilai.

Saat menulis kueri yang berisi klausa UNWIND, gunakan rekomendasi di atas untuk meningkatkan kinerja.

Contoh skenario 1:

```
UNWIND $ids as x
MATCH(t:ticket {`~id`: x})
```

Dengan parameter:

```
parameters={
  "ids": [1, 2, 3]
}
```

Contoh untuk Skenario 2 adalah untuk menghasilkan daftar node untuk CREATE atau MERGE. Alih-alih mengeluarkan beberapa pernyataan, gunakan pola berikut untuk mendefinisikan properti sebagai sekumpulan peta yang diratakan:

```
UNWIND $props as p
CREATE(t:ticket {title: p.title, severity:p.severity})
```

Dengan parameter:

```
parameters={
  "props": [
    {"title": "food poisoning", "severity": "2"},
    {"title": "Simone is in office", "severity": "3"}
  ]
}
```

Alih-alih objek simpul bersarang seperti:

```
UNWIND $nodes as n
CREATE(t:ticket n.properties)
```

Dengan parameter:

```
parameters={
  "nodes": [
    {"id": "ticket1", "properties": {"title": "food poisoning", "severity": "2"}},
    {"id": "ticket2", "properties": {"title": "Simone is in office", "severity": "3"}}
  ]
}
```

Tempatkan node yang lebih ketat di sisi kiri dalam ekspresi Variable-Length Path (VLP)

Dalam kueri Variable-Length Path (VLP), mesin kueri mengoptimalkan evaluasi dengan memilih untuk memulai traversal di sisi kiri atau kanan ekspresi. Keputusan didasarkan pada kardinalitas pola di sisi kiri dan kanan. Kardinalitas adalah jumlah node yang cocok dengan pola yang ditentukan.

- Jika pola yang tepat memiliki kardinalitas satu, maka sisi kanan akan menjadi titik awal.
- Jika sisi kiri dan kanan memiliki kardinalitas satu, ekspansi diperiksa di kedua sisi dan dimulai di samping dengan ekspansi yang lebih kecil. Ekspansi adalah jumlah tepi keluar atau masuk untuk node di sebelah kiri dan node di sisi kanan ekspresi VLP. Bagian optimasi ini hanya digunakan jika hubungan VLP searah dan jenis hubungan disediakan.

- Jika tidak, sisi kiri akan menjadi titik awal.

Untuk rantai ekspresi VLP, optimasi ini hanya dapat diterapkan pada ekspresi pertama. Yang lain VLPs dievaluasi dimulai dengan sisi kiri. Sebagai contoh, biarkan kardinalitas (a), (b) menjadi satu, dan kardinalitas (c) lebih besar dari satu.

- `(a)-[*1..]->(c)`: Evaluasi dimulai dengan (a).
- `(c)-[*1..]->(a)`: Evaluasi dimulai dengan (a).
- `(a)-[*1..]-(c)`: Evaluasi dimulai dengan (a).
- `(c)-[*1..]-(a)`: Evaluasi dimulai dengan (a).

Sekarang biarkan tepi masuk (a) menjadi dua, dan tepi keluar (a) menjadi tiga, tepi masuk (b) menjadi empat, dan tepi keluar (b) menjadi lima.

- `(a)-[*1..]->(b)`: Evaluasi dimulai dengan (a) karena tepi keluar (a) kurang dari tepi masuk (b).
- `(a)<-[*1..]-(b)`: Evaluasi dimulai dengan (a) karena tepi masuk (a) kurang dari tepi keluar (b).

Sebagai aturan umum, tempatkan pola yang lebih ketat di sisi kiri ekspresi VLP.

Hindari pemeriksaan label node yang berlebihan dengan menggunakan nama hubungan granular

Saat mengoptimalkan kinerja, menggunakan label hubungan yang eksklusif untuk pola simpul memungkinkan penghapusan pemfilteran label pada node. Pertimbangkan model grafik di mana hubungan hanya `likes` digunakan untuk mendefinisikan hubungan antara dua `person` node. Kita bisa menulis query berikut untuk menemukan pola ini:

```
MATCH (n:person)-[:likes]->(m:person)
RETURN n, m
```

Pemeriksaan `person` label pada n dan m berlebihan, karena kami mendefinisikan hubungan hanya muncul ketika keduanya bertipe. `person` Untuk mengoptimalkan kinerja, kita dapat menulis kueri sebagai berikut:

```
MATCH (n)-[:likes]->(m)
```

```
RETURN n, m
```

Pola ini juga dapat diterapkan ketika properti eksklusif untuk label node tunggal. Asumsikan bahwa hanya `person` node yang memiliki properti`email`, oleh karena itu memverifikasi kecocokan `person` label node berlebihan. Menulis kueri ini sebagai:

```
MATCH (n:person)
WHERE n.email = 'xxx@gmail.com'
RETURN n
```

Kurang efisien daripada menulis kueri ini sebagai:

```
MATCH (n)
WHERE n.email = 'xxx@gmail.com'
RETURN n
```

Anda hanya harus mengadopsi pola ini ketika kinerja penting dan Anda memiliki pemeriksaan dalam proses pemodelan Anda untuk memastikan label tepi ini tidak digunakan kembali untuk pola yang melibatkan label simpul lainnya. Jika Anda kemudian memperkenalkan `email` properti pada label node lain seperti`company`, maka hasilnya akan berbeda antara dua versi kueri ini.

Tentukan label tepi jika memungkinkan

Disarankan untuk memberikan label tepi jika memungkinkan saat menentukan tepi dalam suatu pola. Pertimbangkan contoh pertanyaan berikut, yang digunakan untuk menghubungkan semua orang yang tinggal di kota dengan semua orang yang mengunjungi kota itu.

```
MATCH (person)-->(city {country: "US"})-->(anotherPerson)
RETURN person, anotherPerson
```

Jika model grafik Anda menautkan orang ke node selain hanya kota yang menggunakan beberapa label tepi, dengan tidak menentukan label akhir, Neptunus perlu mengevaluasi jalur tambahan yang nantinya akan dibuang. Dalam kueri di atas, karena label tepi tidak diberikan, mesin melakukan lebih banyak pekerjaan terlebih dahulu dan kemudian menyaring nilai untuk mendapatkan hasil yang benar. Versi yang lebih baik dari kueri di atas mungkin:

```
MATCH (person)-[:livesIn]->(city {country: "US"})-[:visitedBy]->(anotherPerson)
RETURN person, anotherPerson
```

Ini tidak hanya membantu dalam evaluasi, tetapi memungkinkan perencana kueri untuk membuat rencana yang lebih baik. Anda bahkan dapat menggabungkan praktik terbaik ini dengan pemeriksaan label node redundan untuk menghapus pemeriksaan label kota dan menulis kueri sebagai:

```
MATCH (person)-[:livesIn]->({country: "US"})-[:visitedBy]->(anotherPerson)
RETURN person, anotherPerson
```

Hindari menggunakan klausa WITH jika memungkinkan

Klausa WITH di OpenCypher bertindak sebagai batas di mana segala sesuatu sebelum dijalankan, dan kemudian nilai yang dihasilkan diteruskan ke bagian yang tersisa dari kueri. Klausa WITH diperlukan ketika Anda memerlukan agregasi sementara atau ingin membatasi jumlah hasil, tetapi selain itu Anda harus mencoba menghindari penggunaan klausa WITH. Panduan umum adalah menghapus klausa WITH sederhana ini (tanpa agregasi, urutan berdasarkan atau batas) untuk memungkinkan perencana kueri bekerja pada seluruh kueri untuk membuat rencana optimal secara global. Sebagai contoh, asumsikan Anda menulis kueri untuk mengembalikan semua orang yang tinggal di`India`:

```
MATCH (person)-[:lives_in]->(city)
WITH person, city
MATCH (city)-[:part_of]->(country {name: 'India'})
RETURN collect(person) AS result
```

Pada versi di atas, klausa WITH membatasi penempatan pola `(city)-[:part_of]->(country {name: 'India'})` (yang lebih membatasi) sebelumnya. `(person)-[:lives_in]->(city)` Ini membuat rencana menjadi kurang optimal. Pengoptimalan pada kueri ini adalah menghapus klausa WITH dan membiarkan perencana menghitung paket terbaik.

```
MATCH (person)-[:lives_in]->(city)
MATCH (city)-[:part_of]->(country {name: 'India'})
RETURN collect(person) AS result
```

Tempatkan filter restriktif sedini mungkin dalam kueri

Dalam semua skenario, penempatan awal filter dalam kueri membantu mengurangi solusi perantara yang harus dipertimbangkan oleh rencana kueri. Ini berarti lebih sedikit memori dan lebih sedikit sumber daya komputasi yang diperlukan untuk mengeksekusi kueri.

Contoh berikut membantu Anda memahami dampak ini. Misalkan Anda menulis kueri untuk mengembalikan semua orang yang tinggal di`India`. Salah satu versi kueri dapat berupa:

```
MATCH (n)-[:lives_in]->(city)-[:part_of]->(country)
WITH country, collect(n.firstName + " "  + n.lastName) AS result
WHERE country.name = 'India'
RETURN result
```

Versi kueri di atas bukanlah cara paling optimal untuk mencapai kasus penggunaan ini. Filter `country.name = 'India'` muncul kemudian dalam pola kueri. Ini pertama-tama akan mengumpulkan semua orang dan di mana mereka tinggal, dan mengelompokkan mereka berdasarkan negara, kemudian menyaring hanya untuk grup untuk`country.name = India`. Cara optimal untuk menanyakan hanya orang yang tinggal di `India` dan kemudian melakukan agregasi pengumpulan.

```
MATCH (n)-[:lives_in]->(city)-[:part_of]->(country)
WHERE country.name = 'India'
RETURN collect(n.firstName + " "  + n.lastName) AS result
```

Aturan umum adalah menempatkan filter sesegera mungkin setelah variabel diperkenalkan.

Periksa secara eksplisit apakah properti ada

Berdasarkan semantik OpenCypher, ketika properti diakses itu setara dengan gabungan opsional dan harus mempertahankan semua baris bahkan jika properti tidak ada. Jika Anda tahu berdasarkan skema grafik Anda bahwa properti tertentu akan selalu ada untuk entitas itu, secara eksplisit memeriksa properti itu untuk keberadaan memungkinkan mesin kueri untuk membuat rencana optimal dan meningkatkan kinerja.

Pertimbangkan model grafik di mana node tipe `person` selalu memiliki propertiname. Alih-alih melakukan ini:

```
MATCH (n:person)
RETURN n.name
```

Verifikasi keberadaan properti secara eksplisit dalam kueri dengan pemeriksaan IS NOT NULL:

```
MATCH (n:person)
```

```
WHERE n.name IS NOT NULL
RETURN n.name
```

Jangan gunakan jalur bernama (kecuali jika diperlukan)

Jalur bernama dalam kueri selalu dikenakan biaya tambahan, yang dapat menambahkan penalti dalam hal latensi dan penggunaan memori yang lebih tinggi. Pertimbangkan kueri berikut:

```
MATCH p = (n)-[:commentedOn]->(m)
WITH p, m, n, n.score + m.score as total
WHERE total > 100
MATCH (m)-[:commentedON]->(o)
WITH p, m, n, distinct(o) as o1
RETURN p, m.name, n.name, o1.name
```

Dalam query di atas, dengan asumsi kita hanya ingin mengetahui properti node, penggunaan path “p” tidak perlu. Dengan menentukan jalur bernama sebagai variabel, operasi agregasi menggunakan DISTINCT akan menjadi mahal baik dari segi waktu maupun penggunaan memori. Versi kueri di atas yang lebih dioptimalkan dapat berupa:

```
MATCH (n)-[:commentedOn]->(m)
WITH m, n, n.score + m.score as total
WHERE total > 100
MATCH (m)-[:commentedON]->(o)
WITH m, n, distinct(o) as o1
RETURN m.name, n.name, o1.name
```

Hindari KUMPULKAN (DISTINCT ())

> **Note**
>
> Pada versi mesin 1.4.7.0, penulisan ulang yang disarankan ini tidak lagi diperlukan.

COLLECT (DISTINCT ()) digunakan setiap kali daftar akan dibentuk berisi nilai-nilai yang berbeda. COLLECT adalah fungsi agregasi, dan pengelompokan dilakukan berdasarkan kunci tambahan yang diproyeksikan dalam pernyataan yang sama. Ketika berbeda digunakan, input dibagi menjadi beberapa potongan di mana setiap potongan menunjukkan satu kelompok untuk reduksi. Kinerja akan terpengaruh karena jumlah kelompok meningkat. Di Neptunus, jauh lebih efisien untuk

melakukan DISTINCT sebelum collecting/forming benar-benar daftar. Hal ini memungkinkan pengelompokan dilakukan langsung pada kunci pengelompokan untuk seluruh potongan.

Pertimbangkan kueri berikut:

```
MATCH (n:Person)-[:commented_on]->(p:Post)
WITH n, collect(distinct(p.post_id)) as post_list
RETURN n, post_list
```

Cara yang lebih optimal untuk menulis kueri ini adalah:

```
MATCH (n:Person)-[:commented_on]->(p:Post)
WITH DISTINCT n, p.post_id as postId
WITH n, collect(postId) as post_list
RETURN n, post_list
```

Lebih suka fungsi properti daripada pencarian properti individu saat mengambil semua nilai properti

`properties()`Fungsi ini digunakan untuk mengembalikan peta yang berisi semua properti untuk entitas, dan jauh lebih efisien daripada mengembalikan properti secara individual.

Dengan asumsi `Person` node Anda berisi 5 properti,`firstName`,`lastName`,`age`,`dept`, dan`company`, kueri berikut akan lebih disukai:

```
MATCH (n:Person)
WHERE n.dept = 'AWS'
RETURN properties(n) as personDetails
```

Daripada menggunakan:

```
MATCH (n:Person)
WHERE n.dept = 'AWS'
RETURN n.firstName, n.lastName, n.age, n.dept, n.company

=== OR ===

MATCH (n:Person)
WHERE n.dept = 'AWS'
```

```
RETURN {firstName: n.firstName, lastName: n.lastName, age: n.age,
department: n.dept, company: n.company} as personDetails
```

Lakukan perhitungan statis di luar kueri

Disarankan untuk menyelesaikan perhitungan statis (mathematical/string operasi sederhana) di sisi klien. Pertimbangkan contoh ini di mana Anda ingin menemukan semua orang satu tahun lebih tua atau kurang dari penulis:

```
MATCH (m:Message)-[:HAS_CREATOR]->(p:person)
WHERE p.age <= ($age + 1)
RETURN m
```

Di sini, `$age` disuntikkan ke dalam query melalui parameter, dan kemudian ditambahkan ke nilai tetap. Nilai ini kemudian dibandingkan dengan`p.age`. Sebaliknya, pendekatan yang lebih baik adalah melakukan penambahan di sisi klien dan meneruskan nilai yang dihitung sebagai parameter $ageplusone. Ini membantu mesin kueri untuk membuat rencana yang dioptimalkan, dan menghindari perhitungan statis untuk setiap baris yang masuk. Mengikuti pedoman ini, versi kueri yang lebih efisien adalah:

```
MATCH (m:Message)-[:HAS_CREATOR]->(p:person)
WHERE p.age <= $ageplusone
RETURN m
```

Masukan batch menggunakan UNWIND alih-alih pernyataan individual

Setiap kali kueri yang sama perlu dijalankan untuk input yang berbeda, alih-alih mengeksekusi satu kueri per input, akan jauh lebih berkinerja untuk menjalankan kueri untuk sekumpulan input.

Jika Anda ingin menggabungkan pada satu set node, salah satu opsi adalah menjalankan kueri gabungan per input:

```
MERGE (n:Person {`~id`: $id})
SET n.name = $name, n.age = $age, n.employer = $employer
```

Dengan parameter:

```
params = {id: '1', name: 'john', age: 25, employer: 'Amazon'}
```

Query di atas perlu dieksekusi untuk setiap masukan. Meskipun pendekatan ini berhasil, mungkin memerlukan banyak kueri untuk dieksekusi untuk satu set input yang besar. Dalam skenario ini, batching dapat membantu mengurangi jumlah kueri yang dieksekusi di server, serta meningkatkan keseluruhan throughput.

Gunakan pola berikut:

```
UNWIND $persons as person
MERGE (n:Person {`~id`: person.id})
SET n += person
```

Dengan parameter:

```
params = {persons: [{id: '1', name: 'john', age: 25, employer: 'Amazon'},
{id: '2', name: 'jack', age: 28, employer: 'Amazon'},
{id: '3', name: 'alice', age: 24, employer: 'Amazon'}...]}
```

Eksperimen dengan ukuran batch yang berbeda disarankan untuk menentukan apa yang terbaik untuk beban kerja Anda.

Lebih suka menggunakan kustom IDs untuk node/hubungan

Neptunus memungkinkan pengguna untuk secara eksplisit menetapkan pada node IDs dan hubungan. ID harus unik secara global dalam kumpulan data dan deterministik agar berguna. ID deterministik dapat digunakan sebagai pencarian atau mekanisme penyaringan seperti properti; Namun, menggunakan ID jauh lebih dioptimalkan dari perspektif eksekusi kueri daripada menggunakan properti. Ada beberapa manfaat menggunakan kustom IDs -

- Properti dapat menjadi nol untuk entitas yang ada, tetapi ID harus ada. Ini memungkinkan mesin kueri untuk menggunakan gabungan yang dioptimalkan selama eksekusi.
- Ketika kueri mutasi bersamaan dijalankan, kemungkinan pengecualian modifikasi bersamaan (CMEs) berkurang secara signifikan ketika digunakan untuk mengakses node karena lebih sedikit kunci IDs yang diambil IDs daripada properti karena keunikannya yang diberlakukan.
- Menggunakan IDs menghindari kemungkinan membuat data duplikat karena Neptunus memberlakukan keunikan pada, tidak seperti properti. IDs

Contoh query berikut menggunakan ID kustom:

> **Note**
>
> Properti `~id` ini digunakan untuk menentukan ID, sedangkan hanya `id` disimpan sebagai properti lainnya.

```
CREATE (n:Person {`~id`: '1', name: 'alice'})
```

Tanpa menggunakan ID khusus:

```
CREATE (n:Person {id: '1', name: 'alice'})
```

Jika menggunakan mekanisme yang terakhir, tidak ada penegakan keunikan dan Anda nantinya dapat menjalankan kueri:

```
CREATE (n:Person {id: '1', name: 'john'})
```

Ini menciptakan node kedua dengan `id=1` bernama`john`. Dalam skenario ini, Anda sekarang akan memiliki dua node dengan`id=1`, masing-masing memiliki nama yang berbeda - (alice dan john).

Hindari melakukan perhitungan ~id dalam kueri

Saat menggunakan kustom IDs dalam kueri, selalu lakukan perhitungan statis di luar kueri dan berikan nilai ini dalam parameter. Ketika nilai statis disediakan, mesin lebih mampu mengoptimalkan pencarian dan menghindari pemindaian dan pemfilteran nilai-nilai ini.

Jika Anda ingin membuat tepi antara node yang ada di database, salah satu opsi dapat berupa:

```
UNWIND $sections as section
MATCH (s:Section {`~id`: 'Sec-' + section.id})
MERGE (s)-[:IS_PART_OF]->(g:Group {`~id`: 'g1'})
```

Dengan parameter:

```
parameters={sections: [{id: '1'}, {id: '2'}]}
```

Dalam kueri di atas, `id` bagian sedang dihitung dalam kueri. Karena perhitungannya dinamis, mesin tidak dapat secara statis inline id dan akhirnya memindai semua node bagian. Mesin kemudian

melakukan post-filtering untuk node yang diperlukan. Ini bisa mahal jika ada banyak node bagian dalam database.

Cara yang lebih baik untuk mencapai ini adalah `Sec-` dengan menambahkan id yang diteruskan ke database:

```
UNWIND $sections as section
MATCH (s:Section {`~id`: section.id})
MERGE (s)-[:IS_PART_OF]->(g:Group {`~id`: 'g1'})
```

Dengan parameter:

```
parameters={sections: [{id: 'Sec-1'}, {id: 'Sec-2'}]}
```

Memperbarui/Menggabungkan beberapa node

Saat menjalankan `MERGE` atau `CREATE` kueri pada beberapa node, disarankan untuk menggunakan kombinasi dengan klausa tunggal versus menggunakan MERGE/CREATE MERGE/CREATE klausa untuk setiap node. `UNWIND` Kueri yang menggunakan satu klausa untuk satu node menyebabkan rencana eksekusi yang tidak efisien karena setiap baris memerlukan pengoptimalan. Ini mengarah ke sebagian besar waktu eksekusi kueri yang dihabiskan dalam pemrosesan statis alih-alih pembaruan yang sebenarnya.

Satu Klausul per node tidak optimal karena tidak menskalakan dengan meningkatnya jumlah node:

```
MERGE (p1:Person {name: 'NameA'})
ON CREATE SET p1 += {prop1: 'prop1V1', prop2: 'prop2V1'}
MERGE (p2:Person {name: 'NameB'})
ON CREATE SET p2 += {prop1: 'prop1V2', prop2: 'prop2V2'}
MERGE (p3:Person {name: 'NameC'})
ON CREATE SET p3 += {prop1: 'prop1V3', prop2: 'prop1V3'}
```

Menggunakan `UNWIND` dalam hubungannya dengan satu MERGE/CREATE klausa memungkinkan untuk perilaku yang sama tetapi rencana eksekusi yang lebih optimal. Dengan mengingat hal ini, kueri yang diubah akan terlihat seperti:

```
## If not using custom id for nodes/relationship
UNWIND [{name: 'NameA', prop1: 'prop1V1', prop2: 'prop2V1'}, {name: 'NameB', prop1:
 'prop1V2', prop2: 'prop2V2'}, {name: 'NameC', prop1: 'prop1V3', prop2: 'prop1V3'}] AS
 props
```

```
MERGE (p:Person {name: props.name})
ON CREATE SET p = props

## If using custom id for nodes/relationship
UNWIND [{`~id`: '1', 'name': 'NameA', 'prop1: 'prop1V1', prop2: 'prop2V1'}, {`~id`:
 '2', name: 'NameB', prop1: 'prop1V2', prop2: 'prop2V2'}, {`~id`: '3', name: 'NameC',
 prop1: 'prop1V3', prop2: 'prop1V3'}] AS props
MERGE (p:Person {`~id`: props.id})
ON CREATE SET p = removeKeyFromMap(props, '~id')
```

Praktik Terbaik Neptune Menggunakan SPARQL

Ikuti praktik terbaik ini saat menggunakan bahasa kueri SPARQL dengan Neptune. Untuk informasi tentang menggunakan SPARQL di Neptune, lihat Mengakses grafik Neptune dengan SPARQL.

Topik

- Mengajukan Kueri Semua Grafik Bernama secara Default
- Menentukan Grafik Bernama yang Akan Dimuat
- Memilih Antara FILTER, FILTER...IN, dan VALUES di Kueri Anda

Mengajukan Kueri Semua Grafik Bernama secara Default

Amazon Neptune mengasosiasikan setiap tripel dengan grafik bernama. Grafik default ditetapkan sebagai gabungan dari semua grafik bernama.

Jika Anda mengirimkan kueri SPARQL tanpa secara eksplisit menentukan grafik melalui kata kunci `GRAPH` atau konstruksi seperti `FROM NAMED`, Neptune selalu menganggap semua tripel dalam instans DB Anda. Misalnya, kueri berikut mengembalikan semua tripel dari titik akhir Neptune SPARQL:

```
SELECT * WHERE { ?s ?p ?o }
```

Tripel yang muncul di lebih dari satu grafik dikembalikan hanya sekali.

Untuk informasi tentang spesifikasi grafik default, lihat bagian Set Data RDF dari spesifikasi Bahasa Kueri SPARQL 1.1.

Menentukan Grafik Bernama yang Akan Dimuat

Amazon Neptune mengasosiasikan setiap tripel dengan grafik bernama. Jika Anda tidak menentukan grafik bernama ketika memuat, memasukkan, atau memperbarui tripel, Neptune menggunakan grafik bernama fallback yang ditentukan oleh URI, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Jika Anda menggunakan pemuat massal Neptune, Anda dapat menentukan grafik bernama untuk digunakan untuk semua triple (atau quad dengan posisi keempat kosong) menggunakan parameter `parserConfiguration: namedGraphUri`. Untuk informasi tentang sintaks perintah `Load` pemuat Neptune, lihat the section called “Perintah Loader”.

Memilih Antara FILTER, FILTER...IN, dan VALUES di Kueri Anda

Ada tiga cara dasar untuk menyuntikkan nilai-nilai dalam kueri SPARQL: `FILTER`, `FILTER...IN`, dan `VALUES`.

Misalnya, anggaplah Anda ingin mencari teman dari beberapa orang dalam satu kueri. Menggunakan `FILTER`, Anda mungkin menyusun kueri Anda sebagai berikut:

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
WHERE {?s foaf:knows ?o. FILTER (?s = ex:person1 || ?s = ex:person2)}
```

Kueri ini mengembalikan semua tripel dalam grafik yang memiliki `?s` yang terikat ke `ex:person1` atau `ex:person2` dan memiliki edge keluar berlabel `foaf:knows`.

Anda juga dapat membuat kueri menggunakan `FILTER...IN` yang mengembalikan hasil yang setara:

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
WHERE {?s foaf:knows ?o. FILTER (?s IN (ex:person1, ex:person2))}
```

Anda juga dapat membuat kueri menggunakan `VALUES` yang dalam kasus ini juga mengembalikan hasil yang setara:

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
WHERE {?s foaf:knows ?o. VALUES ?s {ex:person1 ex:person2}}
```

Meskipun dalam banyak kasus kueri ini secara semantik setara, ada beberapa kasus di mana kedua varian `FILTER` berbeda dari varian `VALUES`:

- Kasus pertama adalah ketika Anda menyuntikkan nilai duplikat, seperti menyuntik orang yang sama dua kali. Dalam hal ini, kueri `VALUES` menyertakan duplikat dalam hasil Anda. Anda dapat secara eksplisit menghilangkan duplikat tersebut dengan menambahkan `DISTINCT` ke klausul `SELECT`. Namun mungkin ada situasi saat Anda benar-benar ingin duplikat dalam hasil kueri untuk injeksi nilai berulang.

  Namun, versi `FILTER` dan `FILTER...IN` mengekstrak nilai hanya sekali ketika nilai yang sama muncul beberapa kali.
- Kasus kedua terkait dengan fakta bahwa `VALUES` selalu melakukan pencocokan yang tepat, sedangkan `FILTER` mungkin menerapkan jenis promosi dan melakukan pencocokan fuzzy dalam beberapa kasus.

  Misalnya, ketika Anda menyertakan literal seperti `"2.0"^^xsd:float` di klausa nilai-nilai Anda, kueri `VALUES` sama persis dengan literal ini, termasuk nilai literal dan tipe data.

  Sebaliknya, `FILTER` menghasilkan kecocokan fuzzy untuk literal numerik ini. Pertandingan dapat mencakup literal dengan nilai yang sama tetapi tipe data numerik yang berbeda, seperti `xsd:double`.

  > **Note**
  >
  > Tidak ada perbedaan antara `VALUES` perilaku `FILTER` dan saat menghitung literal string atau. URIs

Perbedaan antara `FILTER` dan `VALUES` dapat mempengaruhi optimasi dan strategi evaluasi kueri yang dihasilkan. Kecuali kasus penggunaan Anda memerlukan pencocokan fuzzy, sebaiknya gunakan `VALUES` karena menghindari pencarian kasus khusus yang berkaitan dengan casting jenis.

Akibatnya, VALUES sering menghasilkan kueri yang lebih efisien yang berjalan lebih cepat dan lebih murah.

Batas Amazon Neptune

Wilayah

Amazon Neptunus tersedia di Wilayah berikut: AWS

- US East (N. Virginia): `us-east-1`
- AS Timur (Ohio): `us-east-2`
- US West (N. California): `us-west-1`
- US West (Oregon): `us-west-2`
- Canada (Central): `ca-central-1`
- Kanada Barat (Calgary): `ca-west-1`
- South America (São Paulo): `sa-east-1`
- Eropa (Stockholm): `eu-north-1`
- Eropa (Spanyol): `eu-south-2`
- Eropa (Irlandia): `eu-west-1`
- Eropa (London): `eu-west-2`
- Eropa (Paris): `eu-west-3`
- Eropa (Frankfurt): `eu-central-1`
- Timur Tengah (Bahrain): `me-south-1`
- Timur Tengah (UEA): `me-central-1`
- Israel (Tel Aviv): `il-central-1`
- Afrika (Cape Town): `af-south-1`
- Asia Pasifik (Hong Kong): `ap-east-1`
- Asia Pacific (Tokyo): `ap-northeast-1`
- Asia Pasifik (Seoul): `ap-northeast-2`
- Asia Pasifik (Osaka): `ap-northeast-3`
- Asia Pacific (Singapore): `ap-southeast-1`
- Asia Pacific (Sydney): `ap-southeast-2`
- Asia Pasifik (Jakarta): `ap-southeast-3`
- Asia Pasifik (Melbourne): `ap-southeast-4`

- Asia Pasifik (Malaysia): `ap-southeast-5`
- Asia Pasifik (Mumbai): `ap-south-1`
- Asia Pasifik (Hyderabad): `ap-south-2`
- Tiongkok (Beijing): `cn-north-1`
- Tiongkok (Ningxia): `cn-northwest-1`
- AWS GovCloud (AS-Barat): `us-gov-west-1`
- AWS GovCloud (AS-Timur): `us-gov-east-1`

Perbedaan di wilayah Tiongkok

Seperti halnya banyak AWS layanan, Amazon Neptunus beroperasi sedikit berbeda di Tiongkok (Beijing) dan China (Ningxia) daripada di wilayah lain. AWS

Misalnya, ketika Neptune ML menggunakan Amazon API Gateway untuk membuat layanan ekspor, autentikasi IAM diaktifkan secara default. Di wilayah Tiongkok, proses untuk mengubah opsi tersebut yang sedikit berbeda dari di wilayah lain.

Perbedaan ini dan lainnya dijelaskan di sini.

Perbedaan di wilayah Asia Pasifik (Hyderabad)

Fitur Neptunus berikut tidak didukung di wilayah Asia Pasifik (Hyderabad) (): `ap-south-2`

- Database global Neptunus
- Penyalinan snapshot lintas wilayah (lihat) Menyalin Snapshot

Ukuran maksimum volume cluster penyimpanan

Volume cluster Neptunus dapat tumbuh hingga ukuran maksimum 128 tebibytes (TiB) di semua wilayah yang didukung. Lihat Penyimpanan, keandalan, dan ketersediaan Amazon Neptunus.

Ukuran instans DB didukung

Neptunus mendukung kelas instans DB yang berbeda di Wilayah yang berbeda. AWS Untuk mengetahui kelas apa saja yang didukung di Wilayah tertentu, lihat Harga Amazon Neptune dan pilih Wilayah yang Anda minati.

Batas untuk setiap AWS akun

Untuk fitur pengelolaan tertentu, Amazon Neptune menggunakan teknologi operasional yang dibagi dengan Amazon Relational Database Service (Amazon RDS).

Setiap AWS akun memiliki batasan untuk setiap Wilayah pada jumlah sumber daya Amazon Neptunus dan Amazon RDS yang dapat Anda buat. Sumber daya ini termasuk instans DB dan klaster DB.

Setelah batas sumber daya tercapai, panggilan tambahan untuk membuat sumber daya itu gagal dengan pengecualian.

Untuk daftar batas yang digunakan bersama antara Amazon Neptune dan Amazon RDS, lihat Batas di Amazon RDS dalam Panduan Pengguna Amazon RDS.

Koneksi ke Neptunus membutuhkan VPC

Amazon Neptune adalah layanan virtual private cloud (VPC) saja.

Selain itu, instans tidak memungkinkan akses dari luar VPC.

Neptune membutuhkan SSL

Memulai dengan versi mesin `1.0.4.0`, Amazon Neptune hanya memungkinkan koneksi Secure Sockets Layer (SSL) melalui HTTPS untuk setiap instans atau titik akhir klaster.

Neptune membutuhkan TLS versi 1.2, menggunakan rangkaian cipher kuat berikut ini:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

Availability zone dan grup subnet DB

Amazon Neptunus memerlukan grup subnet DB untuk setiap cluster yang memiliki subnet di setidaknya dua Availability Zones () yang didukung. AZs

Kami merekomendasikan menggunakan tiga subnet atau lebih di Availability Zone yang berbeda.

Maksimum payload permintaan HTTP (150 MB)

Ukuran total permintaan HTTP Gremlin dan SPARQL harus kurang dari 150 MB. Jika permintaan melebihi ukuran ini, Neptune mengembalikan `HTTP 400: BadRequestException`.

Batas ini tidak berlaku untuk koneksi Gremlin WebSockets .

Perbedaan implementasi Gremlin

Implementasi Amazon Neptune Gremlin memiliki rincian implementasi khusus yang mungkin berbeda dari implementasi Gremlin lainnya.

Untuk informasi selengkapnya, lihat Kepatuhan standar Gremlin di Amazon Neptune.

Neptunus tidak mendukung karakter nol dalam data string

Neptunus tidak mendukung karakter nol dalam string. Hal ini berlaku dalam data grafik properti untuk Gremlin dan OpenCypher, dan untuk data. RDF/SPARQL

SPARQL UPDATE LOAD dari URI

`SPARQL UPDATE LOAD` dari URI bekerja hanya dengan sumber daya yang berada dalam VPC yang sama.

Ini termasuk Amazon S3 URLs di Wilayah yang sama dengan cluster dengan titik akhir VPC Amazon S3 yang dibuat.

URL Amazon S3 harus HTTPS, dan autentikasi harus disertakan dalam URL. Untuk informasi lebih lanjut, lihat Permintaan autentikasi: Menggunakan Parameter Kueri) dalam Referensi API Layanan Penyimpanan Sederhana Amazon.

Untuk informasi tentang cara membuat VPC Endpoint, lihat Membuat VPC Endpoint Amazon S3.

Jika Anda perlu memuat data dari file, sebaiknya gunakan API loader Amazon Neptune. Untuk informasi selengkapnya, lihat Menggunakan pemuat massal Amazon Neptunus untuk menelan data.

> **Note**
>
> API loader Amazon Neptune adalah non-ACID.

Autentikasi IAM dan kontrol akses

Dalam versi mesin Neptunus sebelum rilis 1.2.0.0, autentikasi IAM dan kontrol akses hanya didukung pada tingkat cluster DB. Namun, dari rilis `1.2.0.0` ke depan, Anda dapat mengontrol akses berbasis kueri pada tingkat yang lebih terperinci menggunakan kunci kondisi dalam kebijakan IAM. Untuk informasi selengkapnya, lihat Menggunakan tindakan kueri dalam pernyataan kebijakan akses data Neptunus dan Mengautentikasi database Amazon Neptunus Anda dengan AWS Identity and Access Management

Konsol Amazon Neptunus memerlukan izin. NeptuneReadOnlyAccess Anda dapat membatasi akses ke pengguna IAM dengan mencabut akses ini. Untuk informasi selengkapnya, lihat Menggunakan kebijakan AWS terkelola untuk mengakses database Amazon Neptunus

Amazon Neptune tidak mendukung kontrol akses berbasis nama pengguna/kata sandi.

WebSocket koneksi bersamaan dan waktu koneksi maksimum

Ada batasan jumlah WebSocket koneksi bersamaan per instans DB Neptunus. Ketika batas itu tercapai, Neptunus membatasi permintaan apa pun untuk membuka koneksi WebSocket baru untuk mencegah penggunaan semua memori heap yang dialokasikan.

Untuk semua jenis instans yang lebih besar yang didukung oleh Neptunus dan semua instans tanpa server, jumlah maksimum koneksi bersamaan WebSocket adalah 32K (32.768).

WebSocket Koneksi bersamaan maksimum untuk tipe instans yang lebih kecil tercantum dalam tabel di bawah ini:

Tipe Instans	Koneksi bersamaan WebSocket maksimum
nirserver	32768

Tipe Instans	Koneksi bersamaan WebSocket maksimum
db.r4.4xlarge	16384
db.r4.2xlarge	8192
db.r4.xlarge	4096
db.r4.large	2048
db.r5.4xlarge	16384
db.r5.2xlarge	8192
db.r5.xlarge	4096
db.r5.large	2048
db.r5d.8xlarge	32768
db.r5d.4xlarge	16384
db.r5d.2xlarge	8192
db.r5d.xlarge	4096
db.r5d.large	2048
db.t3.medium	512
db.t4g.medium	512
db.r6g.4xlarge	16384
db.r6g.2xlarge	8192
db.r6g.xlarge	4096
db.r6g.large	2048
db.r6gd.4xlarge	16384

Tipe Instans	Koneksi bersamaan WebSocket maksimum
db.r6gd.2xlarge	8192
db.r6gd.xlarge	4096
db.r6gd.large	2048
db.x2iezn.2xlarge	16384
db.x2iedn.xlarge	4096
db.x2gd.xlarge	4096
db.x2gd.large	2048
db.x2gd.4xlarge	16384
db.x2gd.2xlarge	8192
db.x2g.xlarge	4096
db.x2g.large	2048
db.x2g.4xlarge	16384
db.x2g.2xlarge	8192
db.x1e.xlarge	4096
db.x1e.2xlarge	16384
db.r6i.xlarge	4096
db.r6i.large	2048
db.r6i.2xlarge	8192
db.r7g.4xlarge	16384
db.r7g.2xlarge	8192

Tipe Instans	Koneksi bersamaan WebSocket maksimum
db.r7g.xlarge	4096
db.r7g.large	2048
db.r8g.4xlarge	16384
db.r8g.2xlarge	8192
db.r8g.xlarge	4096
db.r8g.large	2048
db.r7i.large	2048
db.r7i.xlarge	4096
db.r7i.2xlarge	8192

> **Note**
>
> Neptunus tidak lagi R4 mendukung tipe instans.

Ketika klien menutup sambungan dengan benar, penutupan segera tercermin dalam jumlah koneksi terbuka.

Jika klien tidak menutup sambungan, sambungan mungkin ditutup secara otomatis setelah batas waktu siaga 20 sampai 25 menit (batas waktu siaga adalah waktu berlalu sejak pesan terakhir diterima dari klien). Namun, selama waktu siaga tidak tercapai, Neptune membuat sambungan terbuka tanpa batas waktu.

Ketika autentikasi IAM diaktifkan, WebSocket koneksi selalu terputus beberapa menit lebih dari 10 hari setelah dibuat, jika belum ditutup saat itu.

Batas properti dan label

Tidak ada batas pada jumlah vertex dan edge, atau quad RDF Anda dapat miliki dalam grafik.

Juga tidak ada batasan pada jumlah properti atau label yang dapat dimiliki oleh salah satu vertex atau edge.

Ada batas ukuran 55 MB pada ukuran masing-masing properti atau label. Dalam istilah RDF, ini berarti bahwa nilai dalam setiap kolom (S, P, O atau G) dari quad RDF tidak dapat melebihi 55 MB.

Jika Anda perlu mengasosiasikan objek yang lebih besar seperti gambar dengan vertex atau node dalam grafik Anda, Anda dapat menyimpannya sebagai file di Amazon S3 dan menggunakan jalur Amazon S3 sebagai properti atau label.

Batas yang memengaruhi loader massal Neptune

Anda tidak dapat mengantrekan lebih dari 64 pekerjaan beban massal Neptune sekaligus.

Neptune hanya melacak 1.024 pekerjaan beban massal terbaru.

Neptune hanya menyimpan 10.000 rincian kesalahan terakhir per pekerjaan.

Bekerja dengan AWS layanan lain

Anda dapat menggunakan Amazon Neptunus bersama dengan banyak layanan lainnya: AWS

Integrasi Neptunus dengan layanan lain

- AWS Glue— AWS Glue adalah layanan integrasi data tanpa server yang membantu Anda melakukan pekerjaan ekstrak, transformasi, dan pemuatan (ETL) pada data.

  Neptunus menyediakan perpustakaan sumber terbuka neptune-python-utilities,, yang menyederhanakan penggunaan Python dan Gremlin dalam pekerjaan Glue. Neo4j Spark Connector juga didukung untuk menjalankan pekerjaan Scala dan OpenCypher Glue.
- Amazon SageMaker AI — Amazon SageMaker AI adalah platform pembelajaran mesin berfitur lengkap untuk membangun, melatih, dan menerapkan model pembelajaran mesin berkualitas tinggi.

  Neptunus terintegrasi SageMaker dengan AI dalam dua cara utama:

  - Neptunus menyediakan paket Python open-source untuk notebook Jupyter yang dapat ditemukan di proyek notebook grafik Neptunus. GitHub Paket ini berisi satu set sihir Jupyter, notebook tutorial, dan contoh kode yang menyediakan dalam lingkungan pengkodean interaktif di mana Anda dapat belajar tentang teknologi grafik dan Neptunus. Neptunus menyediakan lingkungan yang dikelola sepenuhnya untuk notebook Jupyter yang dihosting SageMaker oleh AI, dan secara otomatis menautkan ke notebook dalam proyek notebook grafik Neptunus sumber terbuka.
  - Fitur Neptune ML memungkinkan untuk membangun dan melatih model machine learning yang berguna pada grafik besar dalam hitungan jam, bukan minggu. Untuk mencapai hal ini, Neptunus ML menggunakan teknologi jaringan saraf grafik (GNN) yang didukung oleh SageMaker Amazon AI dan Deep Graph Library (DGL).
- AWS Lambda— AWS Lambda fungsi memiliki banyak kegunaan dalam aplikasi Neptunus.

  Untuk informasi tentang cara menggunakan fungsi Lambda dengan salah satu driver Gremlin populer dan varian bahasa, serta contoh spesifik fungsi Lambda yang ditulis dalam Java,, dan Python, JavaScript lihat. Menggunakan AWS Lambda fungsi di Amazon Neptunus
- Amazon Athena - Amazon Athena adalah layanan kueri interaktif yang memudahkan untuk menganalisis data di Amazon Simple Storage Service dan sumber data federasi lainnya menggunakan SQL standar.

Neptunus menyediakan konektor ke Athena yang memungkinkan Athena berkomunikasi dengan data Anda yang disimpan di Neptunus.

- AWS Database Migration Service (AWS DMS) — AWS Database Migration Service adalah layanan AWS web yang dapat Anda gunakan untuk memigrasikan data dari satu database ke database lainnya.

  AWS DMS dapat memuat data ke Neptunus dari basis data sumber yang didukung dengan cepat dan aman. Database sumber tetap beroperasi penuh selama migrasi, meminimalkan waktu henti untuk aplikasi yang mengandalkannya.
- AWS Backup— AWS Backup adalah layanan pencadangan yang dikelola sepenuhnya yang memudahkan untuk memusatkan dan mengotomatiskan cadangan data di seluruh AWS layanan di cloud maupun di tempat.

  AWS Backup memungkinkan Anda membuat snapshot periodik otomatis dari cluster Neptunus menggunakan kebijakan perlindungan data terpusat di seluruh layanan yang AWS didukung untuk database, penyimpanan, dan komputasi.
- AWS SDK untuk panda (sebelumnya dikenal sebagai AWS Data Wrangler, atau`awswrangler`), adalah inisiatif python open-source Layanan AWS Profesional yang memperluas kekuatan pustaka analisis data Python ke, AWS menghubungkan dan lebih dari 30 layanan terkait data, `pandas` termasuk Neptunus. AWS `DataFrames` AWS

  Selain SDK, ada juga tutorial tentang cara menggunakannya dengan Neptunus, dan beberapa contoh notebook Neptunus, yaitu Fraud Ring Detection, Synthetic Identity Detection, dan Logistics Analysis.
- Driver JDBC - Driver Neptunus JDBC mendukung kueri OpenCypher, Gremlin, SQL-Gremlin, dan SPARQL.

  Konektivitas JDBC memudahkan untuk terhubung ke Neptunus dengan alat intelijen bisnis (BI) seperti Tableau.

Alat dan utilitas Neptunus

Amazon Neptunus menyediakan sejumlah alat dan utilitas yang dapat menyederhanakan dan mengotomatiskan pekerjaan Anda dengan grafik. Di antaranya adalah sebagai berikut:

Alat Amazon Neptunus

- Utilitas Amazon Neptunus untuk GraphQL - Utilitas Amazon Neptunus untuk GraphQL adalah alat baris perintah Node.js sumber terbuka yang dapat membantu Anda membuat dan memelihara API GraphQL untuk database grafik properti Neptunus. Ini adalah cara tanpa kode untuk membuat resolver GraphQL untuk kueri GraphQL yang memiliki jumlah parameter input variabel dan mengembalikan sejumlah variabel bidang bersarang.
- Nodestream — Nodestream adalah kerangka kerja untuk menangani pemodelan data semantik sebagai grafik. Ini dirancang agar fleksibel dan dapat diperluas, memungkinkan Anda untuk menentukan bagaimana data dikumpulkan dan dimodelkan sebagai grafik. Ini menggunakan pendekatan berbasis pipeline untuk menentukan bagaimana data dikumpulkan dan diproses, dan menyediakan cara untuk menentukan bagaimana grafik harus diperbarui ketika skema berubah.
- Server Kueri MCP Amazon Neptune — Server Model Context Protocol (MCP) untuk Amazon Neptunus yang mendukung Database Neptunus dan Neptunus Analytics dengan menyediakan kemampuan untuk menjalankan kueri grafik (OpenCypher dan Gremlin) serta mengambil skema grafik.
- Server Memori MCP Amazon Neptunus - Server Model Context Protocol (MCP) untuk Amazon Neptunus yang menyediakan memori kepada agen, disimpan dalam grafik pengetahuan terhadap Amazon Neptunus.

Utilitas Amazon Neptunus untuk GraphQL

Utilitas Amazon Neptune untuk GraphQL adalah alat baris perintah Node.js open-source yang dapat membantu Anda membuat dan memelihara API GraphQL untuk database grafik properti Neptunus (belum berfungsi dengan data RDF). Ini adalah cara tanpa kode untuk membuat resolver GraphQL untuk kueri GraphQL yang memiliki jumlah parameter input variabel dan mengembalikan sejumlah variabel bidang bersarang.

Ini telah dirilis sebagai proyek sumber terbuka yang berlokasi di https://github.com/aws/amazon-neptune-for-graphql.

Anda dapat menginstal utilitas menggunakan NPM seperti ini (lihat Instalasi dan Pengaturan untuk detailnya):

```
npm i @aws/neptune-for-graphql -g
```

Utilitas dapat menemukan skema grafik dari grafik properti Neptunus yang ada, termasuk node, tepi, properti, dan kardinalitas tepi. Kemudian menghasilkan skema GraphQL dengan arahan yang diperlukan untuk memetakan tipe GraphQL ke node dan tepi database, dan secara otomatis menghasilkan kode resolver. Kode resolver dirancang untuk meminimalkan latensi dengan hanya mengembalikan data yang diminta oleh kueri GraphQL.

Anda juga dapat memulai dengan skema GraphQL yang ada dan database Neptunus kosong, dan biarkan utilitas menyimpulkan arahan yang diperlukan untuk memetakan skema GraphQL itu ke node dan tepi data yang akan dimuat ke dalam database. Atau, Anda dapat memulai dengan skema GraphQL dan arahan yang telah Anda buat atau modifikasi.

Utilitas ini mampu menciptakan semua AWS sumber daya yang dibutuhkan untuk pipeline, termasuk AWS AppSync API, peran IAM, sumber data, skema, dan resolver, dan fungsi AWS Lambda yang menanyakan Neptunus.

> **Note**
>
> Contoh baris perintah di sini mengasumsikan konsol Linux. Jika Anda menggunakan Windows, ganti garis miring terbalik ('\') di akhir baris dengan tanda sisipan ('^').

Topik

- Menginstal dan menyiapkan utilitas Amazon Neptunus untuk GraphQL
- Memindai data dalam database Neptunus yang ada
- Mulai dari skema GraphQL tanpa arahan
- Bekerja dengan arahan untuk skema GraphQL
- Argumen baris perintah untuk utilitas GraphQL

Menginstal dan menyiapkan utilitas Amazon Neptunus untuk GraphQL

Jika Anda akan menggunakan utilitas dengan database Neptunus yang ada, Anda memerlukannya untuk dapat terhubung ke titik akhir database. Secara default, database Neptunus hanya dapat diakses dari dalam VPC di mana ia berada.

Karena utilitas adalah alat baris perintah Node.js, Anda harus menginstal Node.js (versi 18 atau lebih tinggi) agar utilitas dapat dijalankan. Untuk menginstal Node.js pada instans EC2 di VPC yang sama dengan database Neptunus Anda, ikuti petunjuknya di sini. Contoh ukuran minimum untuk menjalankan utilitas adalah t2.micro. Selama pembuatan instance pilih VPC database Neptunus dari menu pulldown Common Security Groups.

Untuk menginstal utilitas itu sendiri pada instans EC2 atau mesin lokal Anda, gunakan NPM:

```
npm i @aws/neptune-for-graphql -g
```

Anda kemudian dapat menjalankan perintah bantuan utilitas untuk memeriksa apakah itu diinstal dengan benar:

```
neptune-for-graphql --help
```

Anda mungkin juga ingin menginstal AWS CLI untuk mengelola AWS sumber daya.

Memindai data dalam database Neptunus yang ada

Apakah Anda sudah familiar dengan GraphQL atau tidak, perintah di bawah ini adalah cara tercepat untuk membuat GraphQL API. Ini mengasumsikan bahwa Anda telah menginstal dan mengkonfigurasi utilitas Neptunus untuk GraphQL seperti yang dijelaskan di bagian instalasi, sehingga terhubung ke titik akhir database Neptunus Anda.

```
neptune-for-graphql \
  --input-graphdb-schema-neptune-endpoint (your neptune database endpoint):(port
 number) \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (your new GraphQL API name) \
  --output-resolver-query-https
```

Utilitas menganalisis database untuk menemukan skema node, tepi, dan properti di dalamnya. Berdasarkan skema itu, ia menyimpulkan skema GraphQL dengan kueri dan mutasi terkait.

Kemudian membuat AppSync GraphQL API dan sumber daya yang AWS diperlukan untuk menggunakannya. Sumber daya ini mencakup sepasang peran IAM dan fungsi Lambda yang berisi kode resolver GraphQL.

Ketika utilitas selesai, Anda akan menemukan GraphQL API baru di konsol AppSync dengan nama yang Anda tetapkan dalam perintah. Untuk mengujinya, gunakan opsi AppSync Kueri pada menu.

Jika Anda menjalankan perintah yang sama lagi setelah menambahkan lebih banyak data ke database, akan memperbarui kode AppSync API dan Lambda yang sesuai.

Untuk melepaskan semua sumber daya yang terkait dengan perintah, jalankan:

```
neptune-for-graphql \
  --remove-aws-pipeline-name (your new GraphQL API name from above)
```

Mulai dari skema GraphQL tanpa arahan

Anda dapat memulai dari database Neptunus kosong dan menggunakan skema GraphQL tanpa arahan untuk membuat data dan menanyakannya. Perintah di bawah ini secara otomatis membuat AWS sumber daya untuk melakukan ini:

```
neptune-for-graphql \
  --input-schema-file (your GraphQL schema file)
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (name for your new GraphQL API) \
  --create-update-aws-pipeline-neptune-endpoint (your Neptune database endpoint):(port
 number) \
  --output-resolver-query-https
```

File skema GraphQL harus menyertakan jenis skema GraphQL, seperti yang ditunjukkan pada contoh TODO di bawah ini. Utilitas menganalisis skema Anda dan membuat versi diperpanjang berdasarkan tipe Anda. Ini menambahkan kueri dan mutasi untuk node yang disimpan dalam database grafik, dan jika skema Anda memiliki tipe bersarang, itu menambahkan hubungan antara tipe yang disimpan sebagai tepi dalam database.

Utilitas membuat AppSync GraphQL API, dan semua AWS sumber daya yang dibutuhkan. Ini termasuk sepasang peran IAM dan fungsi Lambda yang berisi kode resolver GraphQL. Ketika perintah selesai, Anda dapat menemukan GraphQL API baru dengan nama yang Anda tentukan di konsol. AppSync Untuk mengujinya, gunakan Query di AppSync menu.

Contoh di bawah ini menggambarkan cara kerjanya:

Contoh todo, mulai dari skema GraphQL tanpa arahan

Dalam contoh ini kita mulai dari skema Todo GraphQL tanpa arahan, yang dapat Anda temukan di direktori. *???samples???* Ini mencakup dua jenis ini:

```
type Todo {
  name: String
  description: String
  priority: Int
  status: String
  comments: [Comment]
}

type Comment {
  content: String
}
```

Perintah ini memproses skema Todo dan titik akhir database Neptunus kosong untuk membuat GraphQL API di: AWS AppSync

```
neptune-for-graphql /
  --input-schema-file ./samples/todo.schema.graphql \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name TodoExample \
  --create-update-aws-pipeline-neptune-endpoint (empty Neptune database endpoint):(port
 number) \
  --output-resolver-query-https
```

Utilitas membuat file baru di folder output yang disebut`TodoExample.source.graphql`, dan GraphQL API di. AppSync Utilitas menyimpulkan hal berikut:

- Pada tipe Todo ditambahkan `@relationship` untuk CommentEdge tipe baru. Ini menginstruksikan resolver untuk menghubungkan Todo ke Komentar menggunakan tepi database grafik yang disebut. CommentEdge
- Ia menambahkan input baru yang dipanggil TodoInput untuk membantu kueri dan mutasi.
- Ini menambahkan dua kueri untuk setiap jenis (Todo, Komentar): satu untuk mengambil satu jenis menggunakan `id` atau salah satu bidang jenis yang tercantum dalam input, dan yang lainnya untuk mengambil beberapa nilai, disaring menggunakan input untuk jenis itu.

- Ini menambahkan tiga mutasi untuk setiap jenis: buat, perbarui, dan hapus. Jenis yang akan dihapus ditentukan menggunakan `id` atau input untuk jenis itu. Mutasi ini mempengaruhi data yang disimpan dalam database Neptunus.
- Ini menambahkan dua mutasi untuk koneksi: sambungkan dan hapus. Mereka mengambil sebagai input id node dari dari dan ke simpul yang digunakan oleh Neptunus dan koneksi adalah tepi dalam database.

Penyelesai mengenali kueri dan mutasi berdasarkan namanya, tetapi Anda dapat menyesuaikannya seperti yang ditunjukkan di bawah ini.

Berikut adalah isi dari `TodoExample.source.graphql` file tersebut:

```
type Todo {
  _id: ID! @id
  name: String
  description: String
  priority: Int
  status: String
  comments(filter: CommentInput, options: Options): [Comment] @relationship(type:
 "CommentEdge", direction: OUT)
  bestComment: Comment @relationship(type: "CommentEdge", direction: OUT)
  commentEdge: CommentEdge
}

type Comment {
  _id: ID! @id
  content: String
}

input Options {
  limit: Int
}

input TodoInput {
  _id: ID @id
  name: String
  description: String
  priority: Int
  status: String
}
```

```
type CommentEdge {
  _id: ID! @id
}

input CommentInput {
  _id: ID @id
  content: String
}

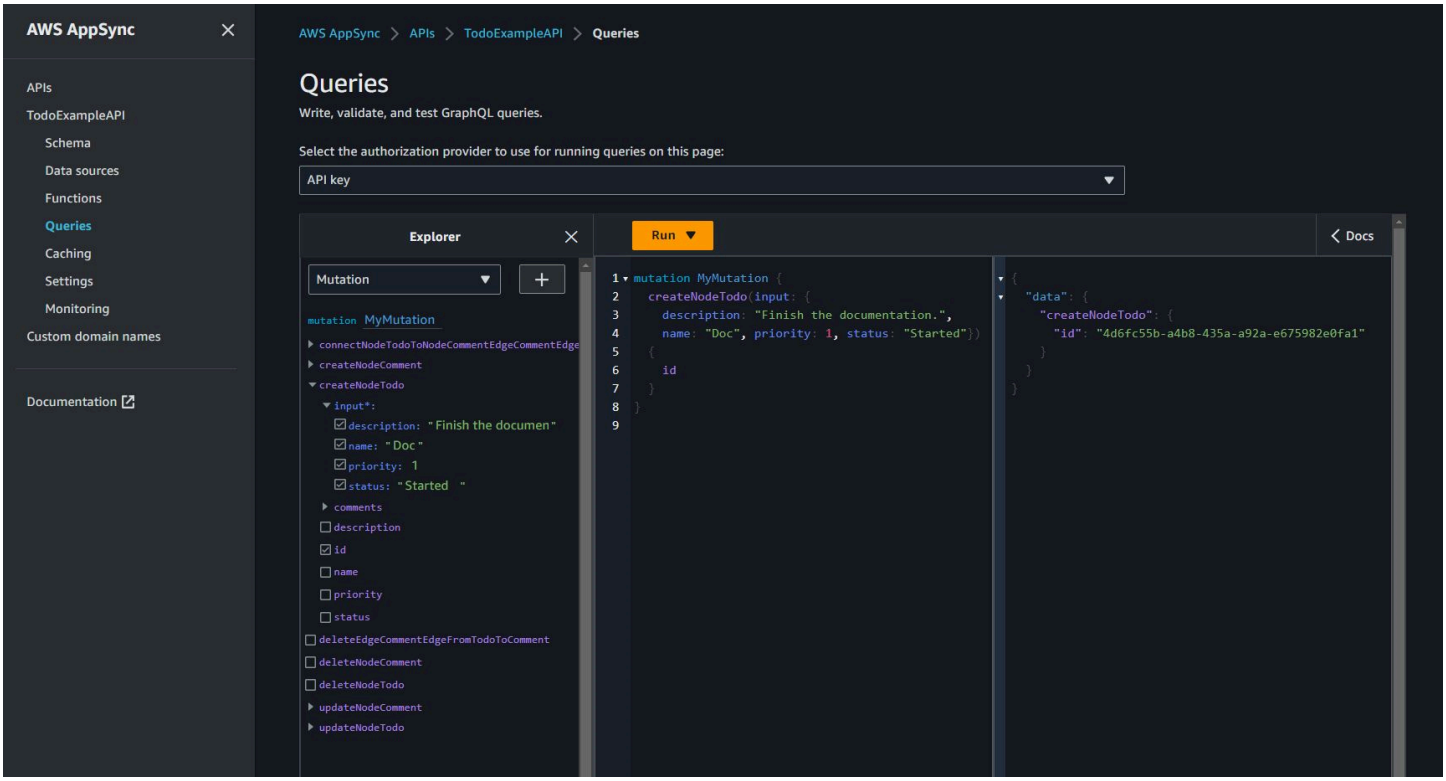
input Options {
  limit: Int
}

type Query {
  getNodeTodo(filter: TodoInput, options: Options): Todo
  getNodeTodos(filter: TodoInput): [Todo]
  getNodeComment(filter: CommentInput, options: Options): Comment
  getNodeComments(filter: CommentInput): [Comment]
}

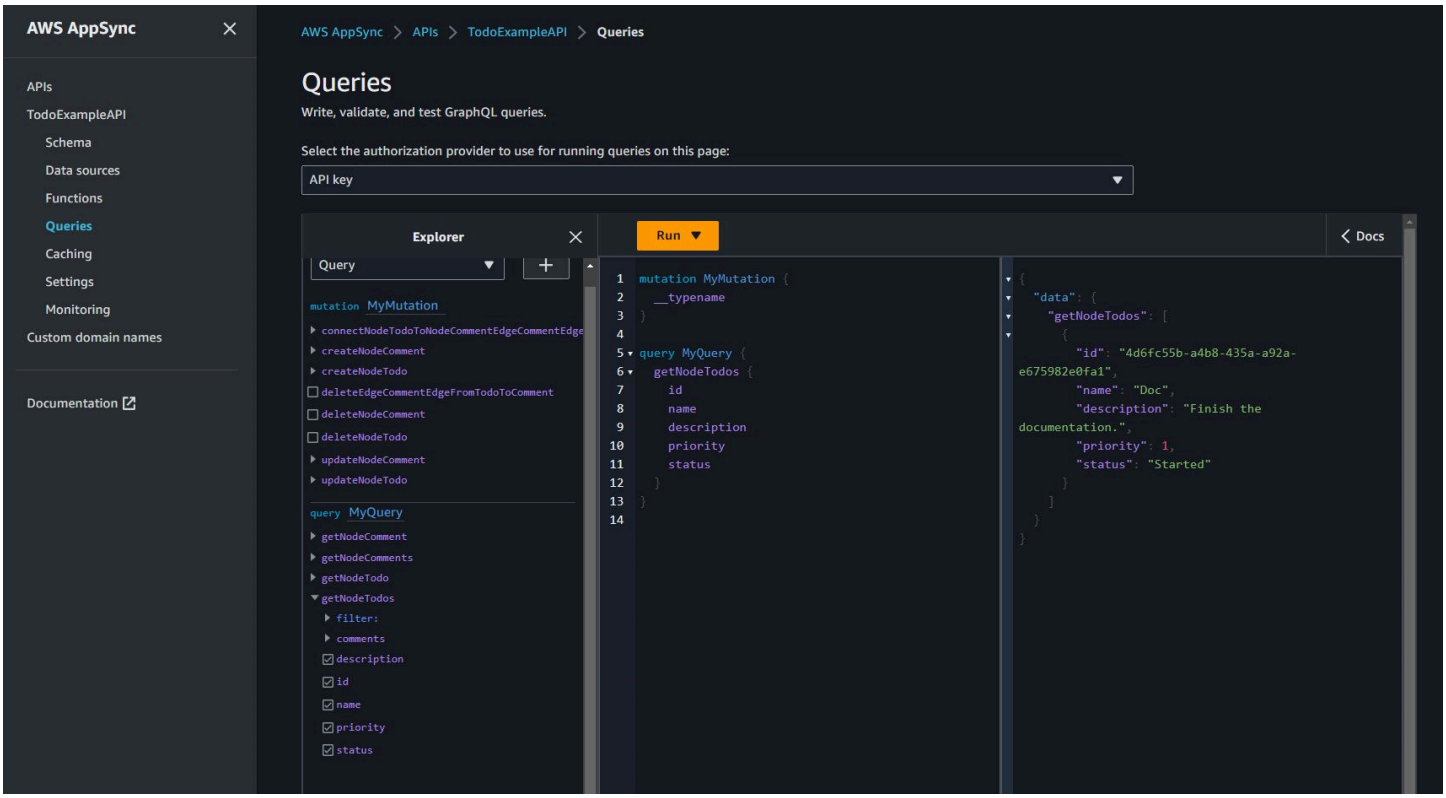
type Mutation {
  createNodeTodo(input: TodoInput!): Todo
  updateNodeTodo(input: TodoInput!): Todo
  deleteNodeTodo(_id: ID!): Boolean
  connectNodeTodoToNodeCommentEdgeCommentEdge(from_id: ID!, to_id: ID!): CommentEdge
  deleteEdgeCommentEdgeFromTodoToComment(from_id: ID!, to_id: ID!): Boolean
  createNodeComment(input: CommentInput!): Comment
  updateNodeComment(input: CommentInput!): Comment
  deleteNodeComment(_id: ID!): Boolean
}

schema {
  query: Query
  mutation: Mutation
}
```

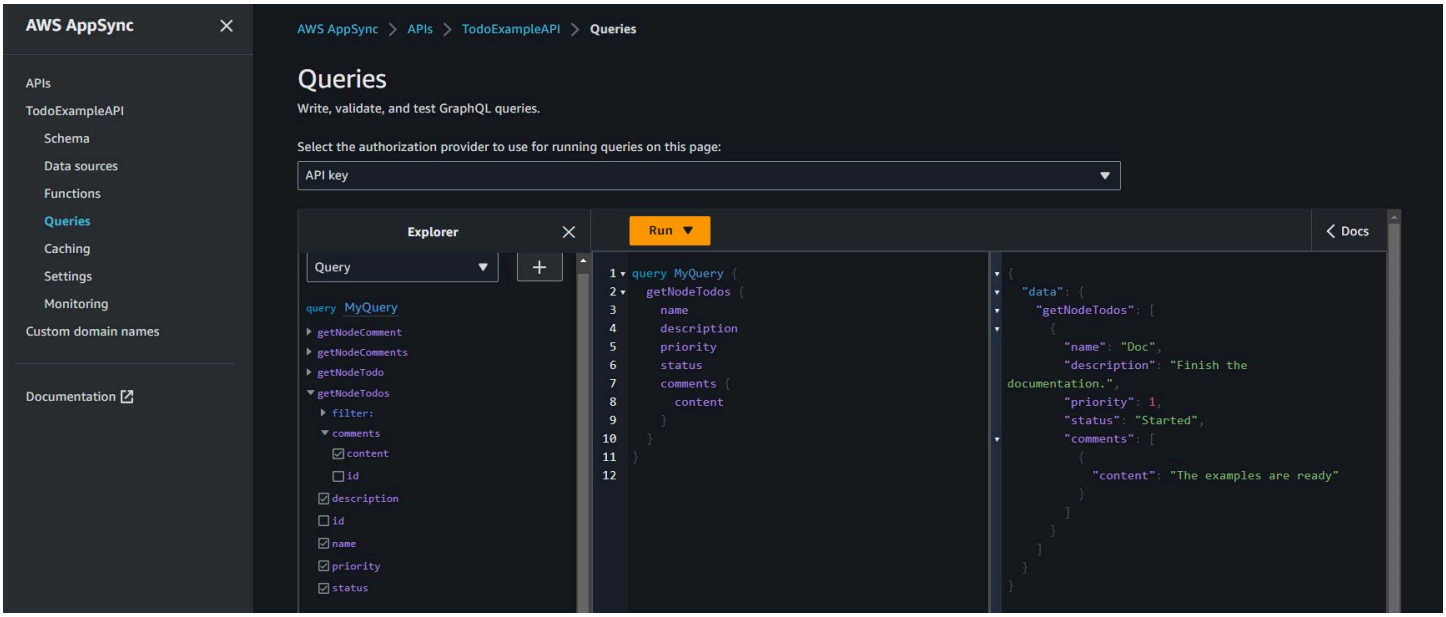
Sekarang Anda dapat membuat dan meminta data. Berikut adalah snapshot dari konsol AppSync Queries yang digunakan untuk menguji GraphQL API baru, yang dinamai dalam kasus ini. `TodoExampleAPI` Di jendela tengah, Explorer menunjukkan daftar kueri dan mutasi dari mana Anda dapat memilih kueri, parameter input, dan bidang pengembalian. Screenshot ini menunjukkan pembuatan tipe node Todo menggunakan `createNodeTodo` mutasi:



Tangkapan layar ini menunjukkan kueri semua node Todo menggunakan kueri: `getNodeTodos`



Setelah membuat Komentar menggunakan`createNodeComment`, Anda dapat menggunakan `connectNodeTodoToNodeCommentEdgeCommentEdge` mutasi untuk menghubungkannya dengan menentukan id mereka. Berikut adalah kueri bersarang untuk mengambil Todos dan komentar terlampirnya:



Jika Anda ingin membuat perubahan pada `TodoExample.source.graphql` file seperti yang dijelaskan dalam Bekerja dengan arahan, Anda kemudian dapat menggunakan skema yang diedit sebagai input dan menjalankan utilitas lagi. Utilitas kemudian akan memodifikasi GraphQL API yang sesuai.

Bekerja dengan arahan untuk skema GraphQL

Anda dapat memulai dari skema GraphQL yang sudah memiliki arahan, menggunakan perintah seperti berikut:

```
neptune-for-graphql \
  --input-schema-file (your GraphQL schema file with directives) \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (name for your new GraphQL API) \
  --create-update-aws-pipeline-neptune-endpoint  (empty Neptune database
 endpoint):(port number) \
  --output-resolver-query-https
```

Anda dapat memodifikasi arahan yang telah dibuat oleh utilitas atau menambahkan arahan Anda sendiri ke skema GraphQL. Berikut adalah beberapa cara untuk bekerja dengan arahan:

Menjalankan utilitas sehingga tidak menghasilkan mutasi

Untuk mencegah utilitas menghasilkan muatasi di GraphQL API, gunakan opsi dalam perintah. `--output-schema-no-mutations` `neptune-for-graphql`

`@alias`Arahan

`@alias`Direktif dapat diterapkan ke jenis atau bidang skema GraphQL. Ini memetakan nama yang berbeda antara database grafik dan skema GraphQL. Sintaksnya adalah:

```
@alias(property: (property name))
```

Dalam contoh di bawah ini `airport` adalah label node database grafik yang dipetakan ke tipe GraphQL, `desc` dan merupakan properti node grafik yang dipetakan ke bidang (lihat Contoh `description` Rute Udara): `Airport`

```
type Airport @alias(property: "airport") {
  city: String
  description: String @alias(property: "desc")
}
```

Perhatikan bahwa pemformatan GraphQL standar memerlukan nama tipe Pascal-casing dan nama bidang casing unta.

`@relationship`Arahan

`@relationship`Direktif memetakan jenis GraphQL bersarang untuk membuat grafik tepi basis data. Sintaksnya adalah:

```
@relationship(edgeType: (edge name), direction: (IN or OUT))
```

Berikut adalah contoh perintah:

```
type Airport @alias(property: "airport") {
  ...
  continentContainsIn: Continent @relationship(edgeType: "contains", direction: IN)
  countryContainsIn: Country @relationship(edgeType: "contains", direction: IN)
  airportRoutesOut(filter: AirportInput, options: Options): [Airport]
 @relationship(edgeType: "route", direction: OUT)
  airportRoutesIn(filter: AirportInput, options: Options): [Airport]
 @relationship(edgeType: "route", direction: IN)
```

```
}
```

Anda dapat menemukan `@relationship` arahan dalam contoh Todo dan Contoh Rute Udara.

`@graphQuery`Dan `@cypher` arahan

Anda dapat menentukan kueri OpenCypher untuk menyelesaikan nilai bidang, menambahkan kueri, atau menambahkan mutasi. Misalnya, ini menambahkan `outboundRoutesCount` bidang baru ke `Airport` tipe untuk menghitung rute outboud:

```
type Airport @alias(property: "airport") {
  ...
  outboundRoutesCount: Int @graphQuery(statement: "MATCH (this)-[r:route]->(a) RETURN
 count(r)")
}
```

Berikut contoh kueri dan mutasi baru:

```
type Query {
  getAirportConnection(fromCode: String!, toCode: String!): Airport \
    @cypher(statement: \
      "MATCH (:airport{code: '$fromCode'})-[:route]->(this:airport)-[:route]-
>(:airport{code:'$toCode'})")
}

type Mutation {
  createAirport(input: AirportInput!): Airport @graphQuery(statement: "CREATE
 (this:airport {$input}) RETURN this")
  addRoute(fromAirportCode:String, toAirportCode:String, dist:Int): Route \
    @graphQuery(statement: \
     "MATCH (from:airport{code:'$fromAirportCode'}),
 (to:airport{code:'$toAirportCode'}) \
      CREATE (from)-[this:route{dist:$dist}]->(to) \
      RETURN this")
}
```

Perhatikan bahwa jika Anda menghilangkanRETURN, resolver mengasumsikan kata kunci `this` adalah lingkup kembali.

Anda juga dapat menambahkan kueri atau mututasi menggunakan kueri Gremlin:

```
type Query {
```

```
  getAirportWithGremlin(code:String): Airport \
    @graphQuery(statement: "g.V().has('airport', 'code', '$code').elementMap()")  #
 single node
  getAirportsWithGremlin: [Airport] \
    @graphQuery(statement: "g.V().hasLabel('airport').elementMap().fold()")       #
 list of nodes
  getCountriesCount: Int \
    @graphQuery(statement: "g.V().hasLabel('country').count()")                   #
 scalar
}
```

Pada saat ini kueri Gremlin terbatas pada kueri yang mengembalikan nilai skalar, atau `elementMap()` untuk satu node, atau `elementMap().fold()` untuk daftar node.

`@id`Arahan

`@id`Direktif mengidentifikasi bidang yang dipetakan ke entitas database `id` grafik. Database grafik seperti Amazon Neptunus selalu memiliki `id` keunikan untuk node dan tepi yang ditetapkan selama impor massal atau yang dibuat secara otomatis. Contoh:

```
type Airport {
  _id: ID! @id
  city: String
  code: String
}
```

Jenis cadangan, kueri, dan nama mutasi

Utilitas membuat kueri dan mutasi secara otomatis untuk membuat GraphQL API yang berfungsi. Pola nama-nama ini dikenali oleh resolver dan dicadangkan. Berikut adalah contoh untuk tipe `Airport` dan tipe penghubung`Route`:

`Options`Tipe ini dicadangkan.

```
input Options {
  limit: Int
}
```

Parameter `filter` dan `options` fungsi dicadangkan.

```
type Query {
```

```
  getNodeAirports(filter: AirportInput, options: Options): [Airport]
}
```

Awalan getNode dari nama kueri dicadangkan, dan awalan nama mutasi seperti`createNode`,,,,, `updateNode` `deleteNode` `connectNode`deleteNode, `updateEdge` dan dicadangkan. `deleteEdge`

```
type Query {
  getNodeAirport(id: ID, filter: AirportInput): Airport
  getNodeAirports(filter: AirportInput): [Airport]
}

type Mutation {
  createNodeAirport(input: AirportInput!): Airport
  updateNodeAirport(id: ID!, input: AirportInput!): Airport
  deleteNodeAirport(id: ID!): Boolean
  connectNodeAirportToNodeAirportEdgeRout(from: ID!, to: ID!, edge: RouteInput!): Route
  updateEdgeRouteFromAirportToAirport(from: ID!, to: ID!, edge: RouteInput!): Route
  deleteEdgeRouteFromAirportToAirport(from: ID!, to: ID!): Boolean
}
```

Menerapkan perubahan pada skema GraphQL

Anda dapat memodifikasi skema sumber GraphQL dan menjalankan utilitas lagi, mendapatkan skema terbaru dari database Neptunus Anda. Setiap kali utilitas menemukan skema baru dalam database, itu menghasilkan skema GraphQL baru.

Anda juga dapat mengedit skema sumber GraphQL secara manual dan menjalankan utilitas lagi menggunakan skema sumber sebagai input alih-alih titik akhir basis data Neptunus.

Akhirnya, Anda dapat menempatkan perubahan Anda dalam file menggunakan format JSON ini:

```
[
  {
    "type": "(GraphQL type name)",
    "field": "(GraphQL field name)",
    "action": "(remove or add)",
    "value": "(value)"
  }
]
```

Contoh:

```
[
  {
    "type": "Airport",
    "field": "outboundRoutesCountAdd",
    "action": "add",
    "value":"outboundRoutesCountAdd: Int @graphQuery(statement: \"MATCH (this)-
[r:route]->(a) RETURN count(r)\")"
  },
  {
    "type": "Mutation",
    "field": "deleteNodeVersion",
    "action": "remove",
    "value": ""
  },
  {
    "type": "Mutation",
    "field": "createNodeVersion",
    "action": "remove",
    "value": ""
  }
]
```

Kemudian, saat Anda menjalankan utilitas pada file ini menggunakan `--input-schema-changes-file` parameter dalam perintah, utilitas menerapkan perubahan Anda sekaligus.

Argumen baris perintah untuk utilitas GraphQL

- **`--help, -h`**— Mengembalikan teks bantuan untuk utilitas GraphQL ke konsol.

- **`--input-schema`** ***`(schema text)`***— Skema GraphQL, dengan atau tanpa arahan, untuk digunakan sebagai input.

- **`--input-schema-file`** ***`(file URL)`***— URL file yang berisi skema GraphQL untuk digunakan sebagai input.

- **`--input-schema-changes-file`** ***`(file URL)`***— URL file yang berisi perubahan yang ingin Anda buat pada skema GraphQL. Jika Anda menjalankan utilitas terhadap database

Neptunus beberapa kali, dan juga mengubah skema sumber GraphQL secara manual, mungkin menambahkan kueri khusus, chnages manual Anda akan hilang. Untuk menghindari hal ini, letakkan perubahan Anda dalam file perubahan dan teruskan menggunakan argumen ini.

File perubahan menggunakan format JSON berikut:

```
[
  {
    "type": "(GraphQL type name)",
    "field": "(GraphQL field name)",
    "action": "(remove or add)",
    "value": "(value)"
  }
]
```

Lihat contoh Todo untuk informasi lebih lanjut.

- **`--input-graphdb-schema` *`(schema text)`*** — Alih-alih menjalankan utilitas terhadap database Neptunus, Anda dapat mengekspresikan skema graphdb dalam bentuk teks untuk digunakan sebagai input. Skema graphdb memiliki format JSON seperti ini:

```
{
  "nodeStructures": [
    { "label":"nodelabel1",
      "properties": [
        { "name":"name1", "type":"type1" }
      ]
    },
    { "label":"nodelabel2",
      "properties": [
          { "name":"name2", "type":"type1" }
      ]
    }
  ],
  "edgeStructures": [
    {
      "label":"label1",
      "directions": [
        { "from":"nodelabel1", "to":"nodelabel2", "relationship":"ONE-ONE|ONE-MANY|
MANY-MANY"  }
```

```
      ],
      "properties": [
        { "name":"name1", "type":"type1" }
      ]
    }
  ]
}
```

- **`--input-graphdb-schema-file`** ***(file URL)***— Alih-alih menjalankan utilitas terhadap database Neptunus, Anda dapat menyimpan skema graphdb dalam file untuk digunakan sebagai input. Lihat `--input-graphdb-schema` di atas untuk contoh format JSON untuk file skema graphdb.

- **`--input-graphdb-schema-neptune-endpoint`** ***(endpoint URL)***— Database Neptunus menunjukkan dari mana utilitas harus mengekstrak skema graphdb.

- **`--output-schema-file`** ***(file name)***- Nama file output untuk skema GraphQL. Jika tidak ditentukan, defaultnya adalah`output.schema.graphql`, kecuali nama pipeline telah disetel menggunakan`--create-update-aws-pipeline-name`, dalam hal ini nama file default adalah*`(pipline name)`*`.schema.graphql`.

- **`--output-source-schema-file`** ***(file name)***- Nama file output untuk skema GraphQL dengan arahan. Jika tidak ditentukan, defaultnya adalah`output.source.schema.graphql`, kecuali nama pipeline telah disetel menggunakan`--create-update-aws-pipeline-name`, dalam hal ini nama defaultnya adalah*`(pipeline name)`*`.source.schema.graphql`.

- **`--output-schema-no-mutations`**— Jika argumen ini ada, utilitas tidak menghasilkan mutasi di GraphQL API, hanya kueri.

- **`--output-neptune-schema-file`** ***(file name)***— Nama file output untuk skema Neptunus graphdb yang ditemukan utilitas. Jika tidak ditentukan, defaultnya adalah`output.graphdb.json`,

kecuali nama pipeline telah disetel menggunakan`--create-update-aws-pipeline-name`, dalam hal ini nama file default adalah*`(pipeline name)`*`.graphdb.json`.

- **`--output-js-resolver-file`** ***`(file name)`*** — Nama file output untuk salinan kode resolver. Jika tidak ditentukan, defaultnya adalah`output.resolver.graphql.js`, kecuali nama pipeline telah disetel menggunakan`--create-update-aws-pipeline-name`, dalam hal ini nama file tersebut*`(pipeline name)`*`.resolver.graphql.js`.

  File ini di-zip dalam paket kode yang diunggah ke fungsi Lambda yang menjalankan resolver.

- **`--output-resolver-query-sdk`** — Argumen ini menetapkan bahwa fungsi Lambda utilitas harus menanyakan Neptunus menggunakan SDK data Neptunus, yang telah tersedia dimulai dengan mesin Neptunus versi 1.2.1.0.R5 (ini adalah default). Namun, jika utilitas mendeteksi versi mesin Neptunus yang lebih lama, itu menyarankan untuk menggunakan opsi Lambda HTTPS sebagai gantinya, yang dapat Anda panggil menggunakan argumen. `--output-resolver-query-https`

- **`--output-resolver-query-https`** — Argumen ini menetapkan bahwa fungsi Lambda utilitas harus menanyakan Neptunus menggunakan API HTTPS Neptunus.

- **`--create-update-aws-pipeline`**- Argumen ini memicu pembuatan AWS sumber daya untuk GraphQL API untuk digunakan, termasuk GraphQL API dan AppSync Lambda yang menjalankan resolver.

- **`--create-update-aws-pipeline-name`** ***`(pipeline name)`*** — Argumen ini menetapkan nama untuk pipeline, seperti `pipeline-name` API untuk AppSync atau `pipeline-name` fungsi untuk fungsi Lambda. Jika nama tidak ditentukan, `--create-update-aws-pipeline` gunakan nama `Neptune` database.
- **`--create-update-aws-pipeline-region`** ***`(AWS region)`*** — Argumen ini menetapkan AWS wilayah di mana pipeline untuk GraphQL API dibuat. Jika tidak ditentukan, wilayah default adalah

salah satu `us-east-1` atau wilayah tempat database Neptunus berada, diekstraksi dari titik akhir database.

- **`--create-update-aws-pipeline-neptune-endpoint`** ***`(endpoint URL)`***Argumen ini menetapkan endpoint database Neptunus yang digunakan oleh fungsi Lambda untuk query database. Jika tidak disetel, titik akhir yang ditetapkan oleh `--input-graphdb-schema-neptune-endpoint` digunakan.
- **`--remove-aws-pipeline-name`** ***`(pipeline name)`***— Argumen ini menghapus pipeline yang dibuat menggunakan`--create-update-aws-pipeline`. Sumber daya yang akan dihapus tercantum dalam file bernama*`(pipeline name)`*`.resources.json`.
- **`--output-aws-pipeline-cdk`**Argumen ini memicu pembuatan file CDK yang dapat digunakan untuk membuat AWS sumber daya untuk GraphQL API, termasuk GraphQL API dan fungsi Lambda yang menjalankan AppSync resolver.
- **`--output-aws-pipeline-cdk-neptume-endpoint`** ***`(endpoint URL)`***Argumen ini menetapkan titik akhir database Neptunus yang digunakan oleh fungsi Lambda untuk menanyakan database Neptunus. Jika tidak disetel, titik akhir yang ditetapkan oleh `--input-graphdb-schema-neptune-endpoint` digunakan.
- **`--output-aws-pipeline-cdk-name`** ***`(pipeline name)`***— Argumen ini menetapkan nama pipeline untuk AppSync API dan fungsi nama pipeline Lambda yang akan digunakan. Jika tidak ditentukan, `--create-update-aws-pipeline` gunakan nama database Neptunus.
- **`--output-aws-pipeline-cdk-region`** ***`(AWS region)`***— Ini menetapkan AWS wilayah di mana pipeline untuk GraphQL API dibuat. Jika tidak ditentukan, default ke `us-east-1` atau wilayah di mana database Neptunus berada, diekstraksi dari titik akhir database.
- **`--output-aws-pipeline-cdk-file`** ***`(file name)`***— Ini menetapkan nama file CDK. Jika tidak diatur defaultnya adalah*`(pipeline name)`*`-cdk.js`.

Nodestream

Nodestream adalah kerangka kerja untuk menangani pemodelan data semantik sebagai grafik. Ini dirancang agar fleksibel dan dapat diperluas, memungkinkan Anda untuk menentukan bagaimana data dikumpulkan dan dimodelkan sebagai grafik. Ini menggunakan pendekatan berbasis pipeline untuk menentukan bagaimana data dikumpulkan dan diproses, dan menyediakan cara untuk menentukan bagaimana grafik harus diperbarui ketika skema berubah. Semua ini dilakukan dengan menggunakan file konfigurasi sederhana yang dapat dibaca manusia dalam format yaml. Untuk mencapai hal ini, Nodestream menggunakan sejumlah konsep inti, termasuk pipeline, extractors, transformer, filter, interpreter, interpretasi, dan migrasi.

Dimulai dengan Nodestream 0.12, Amazon Neptunus didukung untuk Database Neptunus dan Neptunus Analytics.

Silakan lihat dokumentasi Nodestream untuk detail tentang cara mengonfigurasi dan menggunakan Nodestream dengan Neptunus: Dukungan Nodestream untuk Amazon Neptunus.

Nodestream dengan Neptunus saat ini mendukung pipa ETL standar serta saluran pipa time to live (TTL). Pipa ETL memungkinkan penyerapan data massal ke Neptunus dari sumber dan format data yang jauh lebih luas daripada yang sebelumnya dimungkinkan di Neptunus termasuk:

- Tagihan Material Perangkat Lunak
- File termasuk CSV, JSON, JSONL, Parket, txt dan yaml
- Kafka
- Athena
- ISTIRAHAT APIs

Nodestream sepenuhnya mendukung otentikasi IAM saat menghubungkan ke Amazon Neptunus, selama kredensialnya dikonfigurasi dengan benar. Lihat panduan kredensial boto3 untuk informasi selengkapnya tentang mengonfigurasi kredensil dengan benar.

Mekanisme TTL Nodestream juga memungkinkan kemampuan baru yang sebelumnya tidak tersedia di Neptunus. Dengan membuat anotasi elemen grafik yang tertelan dengan stempel waktu, Nodestream dapat membuat pipeline yang secara otomatis kedaluwarsa dan menghapus data yang telah melewati masa pakai yang dikonfigurasi.

Kesalahan Layanan Neptune

Amazon Neptune memiliki dua set kesalahan yang berbeda:

- Kesalahan mesin grafik yang diperuntukkan hanya bagi titik akhir klaster DB Neptune.
- Kesalahan yang terkait dengan APIs untuk membuat dan memodifikasi sumber daya Neptunus dengan AWS SDK dan (). AWS Command Line Interface AWS CLI

Topik

- Instans DB tidak cukup tersedia
- Pesan dan Kode Kesalahan Mesin Grafik
- Pesan dan Kode Kesalahan API Manajemen Klaster DB
- Pesan Kesalahan dan Umpan Neptune Loader

Instans DB tidak cukup tersedia

Kesalahan `InsufficientDBInstanceCapacity` dapat muncul saat Anda mencoba membuat, memulai, atau mengubah instans DB. Kesalahan ini juga dapat muncul saat Anda mencoba memulihkan instans DB dari snapshot DB. Saat kesalahan ini muncul, satu penyebab umumnya adalah kelas instans DB tertentu tidak tersedia di Zona Ketersediaan yang diminta. Anda dapat mencoba salah satu hal berikut untuk memecahkan masalahnya:

- Coba kembali permintaan dengan kelas instans DB yang berbeda.
- Coba kembali permintaan dengan Zona Ketersediaan yang berbeda.
- Coba kembali permintaan tanpa menentukan Zona Ketersediaan eksplisit.

Untuk informasi tentang pemecahan masalah kapasitas instans untuk Amazon EC2, lihat Kapasitas instans tidak mencukupi di EC2 Panduan Pengguna Amazon.

Pesan dan Kode Kesalahan Mesin Grafik

Titik akhir Amazon Neptune mengembalikan kesalahan standar untuk Gremlin dan SPARQL ketika ditemui.

Kesalahan yang spesifik untuk Neptune juga dapat dikembalikan dari titik akhir yang sama. Bagian ini mendokumentasikan pesan kesalahan Neptune, kode, dan tindakan yang disarankan.

> **Note**
>
> Kesalahan ini diperuntukkan hanya bagi titik akhir klaster DB Neptune. APIs Untuk membuat dan memodifikasi sumber daya Neptunus dengan AWS SDK AWS CLI dan memiliki serangkaian kesalahan umum yang berbeda. Untuk informasi tentang kesalahan tersebut, lihat the section called “Kesalahan API”.

Format Kesalahan Mesin Grafik

Pesan kesalahan Neptune mengembalikan kode kesalahan HTTP yang relevan dan respons berformat JSON.

```
HTTP/1.1 400 Bad Request
x-amzn-RequestId: LDM6CJP8RMQ1FHKSC1RBVJFPNVV4KQNSO5AEMF66Q9ASUAAJG
Content-Type: application/x-amz-json-1.0
Content-Length: 465
Date: Thu, 15 Mar 2017 23:56:23 GMT

{
    "requestId": "0dbcded3-a9a1-4a25-b419-828c46342e47",
    "code": "ReadOnlyViolationException",
    "detailedMessage": "The request is rejected because it violates some read-only
 restriction, such as a designation of a replica as read-only."
}
```

Kesalahan Kueri Mesin Grafik

Tabel berikut berisi kode kesalahan, pesan, dan status HTTP.

Tabel ini juga menunjukkan apakah tidak mengapa mencoba kembali permintaan. Umumnya, tidak apa-apa mencoba lagi permintaan jika mungkin berhasil pada percobaan baru.

Kode Kesalahan Layanan Neptunus	Status HTTP	Ok untuk mencoba lagi?	Pesan
`AccessDeniedException`	403	Tidak	Otentikasi atau kegagalan otorisasi.
`BadRequestException`	400	Tidak	Permintaan tidak dapat diselesaikan.
`BadRequestException`	400	Tidak	Ukuran permintaan melebihi nilai maksimum yang diizinkan 157286400 byte.
`CancelledByUserException`	500	Ya	Pemrosesan permintaan dibatalkan oleh klien yang berwenang.
`ConcurrentModificationException`	500	Ya	Pemrosesan permintaan tidak berhasil karena konflik modifikasi. Klien harus mencoba kembali permintaan tersebut.
`ConstraintViolationException`	400	Ya	Mesin kueri menemukan, selama pelaksanaan permintaan, bahwa penyelesaian beberapa operasi tidak mungkin dilakukan tanpa melanggar beberapa kendala integritas

Kode Kesalahan Layanan Neptunus	Status HTTP	Ok untuk mencoba lagi?	Pesan
			data, seperti persisten si simpul masuk dan keluar sambil menambahkan tepi. Kondisi seperti itu biasanya diamati jika ada modifikas i bersamaan pada grafik, dan bersifat sementara. Klien harus mencoba kembali permintaan tersebut.
`FailureBy QueryException`	500	Ya	Panggilan `fail()` menyebabkan pemrosesan permintaan gagal. Klien dapat mencoba kembali permintaan dengan asumsi logika dalam kueri dapat diharapkan berhasil pada upaya selanjutn ya. Akibatnya, pilihan untuk mencoba lagi akan menjadi kueri khusus.
`InternalF ailureExc eption`	500	Tidak	Pemrosesan permintaan telah gagal.

Kode Kesalahan Layanan Neptunus	Status HTTP	Ok untuk mencoba lagi?	Pesan
`InvalidNumericDataException`	400	Tidak	Penggunaan data numerik tidak valid yang tidak dapat direpresentasikan dalam ukuran penyimpanan 64-bit.
`InvalidParameterException`	400	Tidak	out-of-rangeNilai yang tidak valid diberikan untuk beberapa parameter input atau sintaks tidak valid dalam file RDF yang disediakan.
`MalformedQueryException`	400	Tidak	Permintaan ditolak karena berisi kueri yang secara sintaksis salah atau tidak lulus validasi tambahan.
`MemoryLimitExceededException`	500	Ya	Pemrosesan permintaan tidak berhasil karena kurangnya memori, tetapi dapat dicoba lagi ketika server tidak terlalu sibuk.
`MethodNotAllowedException`	405	Tidak	Permintaan ditolak karena metode HTTP yang dipilih tidak didukung oleh titik akhir yang digunakan.

Kode Kesalahan Layanan Neptunus	Status HTTP	Ok untuk mencoba lagi?	Pesan
`MissingParameterException`	400	Tidak	Parameter yang diperlukan untuk tindakan tertentu tidak disediakan.
`QueryLimitExceededException`	500	Ya	Pemrosesan permintaan tidak berhasil karena kurangnya sumber daya yang terbatas, tetapi dapat dicoba lagi ketika server kurang sibuk.
`QueryLimitException`	400	Tidak	Ukuran kueri melebihi batas sistem.
`QueryTooLargeException`	400	Tidak	Permintaan itu ditolak karena tubuhnya terlalu besar.
`ReadOnlyViolationException`	400	Tidak	Permintaan ditolak karena melanggar beberapa pembatasan hanya-baca, seperti penunjukan replika sebagai hanya-baca.
`ThrottlingException`	500	Ya	Tingkat permintaan melebihi throughput maksimum. OK untuk mencoba lagi.

Kode Kesalahan Layanan Neptunus	Status HTTP	Ok untuk mencoba lagi?	Pesan
`TimeLimitExceededException`	500	Ya	Waktu pemrosesan permintaan habis.
`TooManyRequestsException`	429	Ya	Tingkat permintaan melebihi throughput maksimum. OK untuk mencoba lagi.
`UnsupportedOperationException`	400	Tidak	Permintaan menggunakan fitur atau konstruksi yang saat ini tidak didukung.

Kesalahan Autentikasi IAM

Kesalahan ini khusus untuk klaster yang memiliki autentikasi IAM diaktifkan.

Tabel berikut berisi kode kesalahan, pesan, dan status HTTP.

Kode Kesalahan Layanan Neptunus	Status HTTP	Pesan
Pengguna/Kebijakan IAM Salah	403	Anda tidak memiliki akses yang memadai untuk melakukan tindakan ini.
Wilayah Salah atau Hilang	403	Kredensi harus dicakup ke Wilayah yang valid, bukan ". *region*
Nama Layanan Salah atau Hilang	403	Kredensi harus dicakup untuk memperbaiki layanan: ". `neptune-db`

Kode Kesalahan Layanan Neptunus	Status HTTP	Pesan
Header Host Salah atau Hilang/Tanda Tangan Tidak Valid	403	Tanda tangan permintaan yang kami hitung tidak sesuai dengan tanda tangan yang Anda berikan. Periksa Kunci Akses AWS Rahasia Anda dan metode penandatanganan. Konsultasikan dokumentasi layanan untuk detailnya. Header host tidak ada atau nama host salah.
Hilang `X-Amz-Security-Token`	403	'`x-amz-security-token` ' dinamai sebagai a`SignedHeader` , tetapi tidak ada dalam permintaan HTTP
Header Otorisasi Hilang	403	Permintaan tidak menyertakan header otorisasi yang diperlukan, atau formatnya salah.
Token Autentikasi Hilang	403	Token Otentikasi Hilang.
Tanggal Lama	403	Tanda tangan kedaluwarsa: *20181011T213907Z* sekarang lebih awal dari *20181011T213915Z* (*20181011T214415Z -5 min*.)
Tanggal Masa Depan	403	Tanda tangan belum terkini: *20500224T213559Z* masih lebih lambat dari *20181108T225925Z* (*20181108T225425Z +5 min*.)

Kode Kesalahan Layanan Neptunus	Status HTTP	Pesan
Format Tanggal Salah	403	Tanggal harus dalam format dasar ISO-8601. Punya '*date*'. Lihat https://en.wikipedia.org/wiki/ISO_8601.
Kunci Akses Tidak Diketahui/Hilang atau Token Sesi	403	Token keamanan yang disertakan dalam permintaan tidak valid.
Kunci Rahasia Tidak Diketahui/Hilang	403	Tanda tangan permintaan yang kami hitung tidak sesuai dengan tanda tangan yang Anda berikan. Periksa Kunci Akses AWS Rahasia Anda dan metode penandatanganan. Konsultasikan dokumentasi layanan untuk detailnya. Header host tidak ada atau nama host salah.
`TooManyRequestsException`	429	Tingkat permintaan melebihi throughput maksimum. OK untuk mencoba lagi.

Pesan dan Kode Kesalahan API Manajemen Klaster DB

Kesalahan Amazon Neptunus ini dikaitkan dengan untuk membuat dan memodifikasi APIs sumber daya Neptunus dengan SDK dan. AWS AWS CLI

Tabel berikut berisi kode kesalahan, pesan, dan status HTTP.

Kode Kesalahan Layanan Neptunus	Status HTTP	Pesan
`AccessDeniedException`	403	Anda tidak memiliki akses yang memadai untuk melakukan tindakan ini.
`IncompleteSignature`	400	Tanda tangan permintaan tidak sesuai dengan AWS standar.
`InternalFailure`	500	Pemrosesan permintaan telah gagal karena kesalahan yang tidak diketahui, pengecualian, atau kegagalan.
`InvalidAction`	400	Tindakan atau operasi yang diminta tidak valid. Verifikasi bahwa tindakan diketik dengan benar.
`InvalidClientTokenId`	403	Sertifikat X.509 atau ID kunci AWS akses yang disediakan tidak ada dalam catatan kami.
`InvalidParameterCombination`	400	Parameter yang tidak boleh digunakan bersama digunakan secara bersamaan.
`InvalidParameterValue`	400	out-of-rangeNilai atau tidak valid diberikan untuk parameter input.
`InvalidQueryParameter`	400	out-of-rangeNilai atau tidak valid diberikan untuk parameter input.
`MalformedQueryString`	400	String kueri berisi kesalahan sintaks.

Kode Kesalahan Layanan Neptunus	Status HTTP	Pesan
`MissingAction`	400	Permintaan tidak memiliki tindakan atau parameter yang diperlukan.
`MissingAuthenticationToken`	403	Permintaan harus berisi ID kunci AWS akses yang valid (terdaftar) atau sertifikat X.509.
`MissingParameter`	400	Parameter yang diperlukan untuk tindakan tertentu tidak disediakan.
`OptInRequired`	403	ID kunci AWS akses memerlukan langganan untuk layanan ini.
`RequestExpired`	400	Permintaan mencapai layanan lebih dari 15 menit setelah cap tanggal pada permintaan atau lebih dari 15 menit setelah tanggal kedaluwarsa permintaan (seperti untuk presigned URLs), atau cap tanggal pada permintaan lebih dari 15 menit di masa depan.
`ServiceUnavailable`	503	Permintaan telah gagal karena kegagalan sementara server.
`ThrottlingException`	500	Permintaan ditolak karena throttling permintaan.

Kode Kesalahan Layanan Neptunus	Status HTTP	Pesan
`ValidationError`	400	Input gagal memenuhi kendala yang ditentukan oleh layanan. AWS

Pesan Kesalahan dan Umpan Neptune Loader

Pesan berikut dikembalikan oleh titik akhir `status` dari Neptune Loader. Untuk informasi selengkapnya, lihat API Get-Status.

Tabel berikut berisi kode umpan dan deskripsi loader.

Kesalahan atau Kode Umpan	Deskripsi
LOAD_NOT_STARTED	Beban telah direkam tetapi tidak dimulai.
LOAD_IN_PROGRESS	Menunjukkan bahwa pemuatan sedang berlangsung dan menentukan jumlah file yang sedang dimuat. Ketika loader mem-parsing file, itu membuat satu atau lebih potongan untuk dimuat secara paralel. Karena satu file dapat menghasilkan beberapa potongan, jumlah file yang disertaka n dengan pesan ini biasanya kurang dari jumlah utas yang digunakan oleh proses pemuatan massal.
LOAD_COMPLETED	Beban telah selesai tanpa kesalahan atau kesalahan dalam ambang batas yang dapat diterima.
LOAD_CANCELLED_BY_USER	Beban telah dibatalkan oleh pengguna.
LOAD_CANCELLED_DUE_TO_ERRORS	Beban telah dibatalkan oleh sistem karena kesalahan.

Kesalahan atau Kode Umpan	Deskripsi
LOAD_UNEXPECTED TED_ERROR	Beban gagal dengan kesalahan tak terduga.
LOAD_FAILED	Beban gagal sebagai akibat dari satu kesalahan atau lebih.
LOAD_S3_READ_ERROR	Umpan gagal karena masalah konektivitas Amazon S3 intermiten atau sementara. Jika salah satu umpan menerima kesalahan ini, status beban keseluruhan diatur ke LOAD_FAIL ED.
LOAD_S3_ACCESS_DENIED_ERROR	Akses ditolak ke bucket S3. Jika salah satu umpan menerima kesalahan ini, status beban keseluruhan diatur ke LOAD_FAILED.
LOAD_COMMITTED_W_WRITE_CONFLICTS	Data yang dimuat berkomitmen dengan konflik penulisan yang belum terselesaikan. Loader akan mencoba menyelesaikan konflik tulis dalam transaksi terpisah dan memperbar ui status umpan saat beban berlangsung. Jika status umpan akhir adalah LOAD_COMM ITTED_W_WRITE_CONFLITS, maka cobalah melanjutkan beban dan kemungkinan akan berhasil tanpa konflik tulis. Konflik tulis biasanya tidak terkait dengan data input yang buruk, tetapi duplikat dalam data dapat meningkatkan kemungkinan konflik tulis.
LOAD_DATA_DEADLOCK	Beban secara otomatis diputar kembali karena kebuntuan.
LOAD_DATA_FAILED_DUE_TO_FEE D_MODIFIED_OR_DELETED	Umpan gagal karena file telah dihapus atau diperbarui setelah beban dimulai.
LOAD_FAILED_BECAUSE_DEPENDE NCY_NOT_SATISFIED	Permintaan beban tidak dijalankan karena pemeriksaan ketergantungan gagal.

Kesalahan atau Kode Umpan	Deskripsi
LOAD_IN_QUEUE	Permintaan beban telah diantrekan dan menunggu untuk dieksekusi.
LOAD_FAILED_INVALID_REQUEST	Pemuatan gagal karena permintaan tidak valid (misalnya, yang ditentukan source/bucket mungkin tidak ada, atau format file tidak valid).

Rilis mesin untuk Amazon Neptunus

Amazon Neptune merilis update mesin secara berkala.

Anda dapat menentukan versi rilis engine mana yang saat ini telah Anda instal menggunakan API instance-status atau konsol Neptunus. Nomor versi memberi tahu Anda apakah Anda menjalankan rilis mayor asli, atau rilis minor, atau rilis patch.. Untuk informasi selengkapnya tentang penomoran rilis, lihat Nomor versi mesin.

Untuk informasi selengkapnya tentang pembaruan secara umum, lihatPemeliharaan cluster.

Dari rilis mesin 1.3.0.0 ke depan, versi mesin akan memiliki struktur yang ditunjukkan pada tabel di bawah ini. Nomor versi minor adalah nomor yang akan dievaluasi untuk `AutoMinorVersionUpgrade`diproses.

Versi	Versi produ	Versi besa	Versi minor	Versi patch	Status	Dirilis	Akhir hidup	Tingkatkan ke:
1.4.7.0	1	4	7	0	aktif	2026-03-03	2027-06-03	N/A
1.4.6.3	1	4	6	3	aktif	2025-12-18	2027-06-03	1.4.7.0
1.4.6.2	1	4	6	2	aktif	2025-11-18	2027-06-03	1.4.7.0
1.4.6.1	1	4	6	1	aktif	2025-09-18	2027-03-06	1.4.6.2
1.4.6.0	1	4	6	0	aktif	2025-09-02	2027-03-06	1.4.6.1
1.4.5.1	1	4	5	1	aktif	2025-06-30	2027-03-06	1.4.6.0
1.4.5.0	1	4	5	0	aktif	2025-04-09	2027-03-06	1.4.5.1

Versi	Versi produ	Versi besa	Versi minor	Versi patch	Status	Dirilis	Akhir hidup	Tingkatkan ke:
1.4.4.0	1	4	4	0	aktif	2025-02-24	2027-03-06	1.4.5.0
1.4.3.0	1	4	3	0	aktif	2025-01-21	2027-03-06	1.4.4.0
1.4.2.0	1	4	2	0	aktif	2024-12-19	2027-03-06	1.4.3.0
1.4.1.0	1	4	1	0	aktif	2024-11-21	2027-03-06	1.4.2.0
1.4.0.0	1	4	0	0	aktif	2024-11-06	2027-03-06	1.4.1.0
1.3.4.0	1	3	4	0	aktif	2024-10-01	2027-03-06	1.4.0.0
1.3.3.0	1	3	3	0	aktif	2024-08-05	2027-03-06	1.3.4.0
1.3.2.1	1	3	2	1	aktif	2024-06-20	2027-03-06	1.3.3.0
1.3.2.0	1	3	2	0	aktif	2024-06-10	2027-03-06	1.3.2.1
1.3.1.0	1	3	1	0	aktif	2024-03-06	2027-03-06	1.3.2.1
1.3.0.0	1	3	0	0	aktif	2023-11-15	2027-03-06	1.3.2.1

Tabel di bawah ini mencantumkan semua rilis mesin sejak 1.0.1.0, bersama dengan informasi tentang versi. end-of-life Anda dapat menggunakan tanggal dalam tabel ini untuk merencanakan siklus pengujian dan peningkatan Anda.

Versi	Versi besar	Versi minor	Status	Dirilis	Akhir hidup	Tingkatkan ke:
1.2.1.2	1.2	1.2	aktif	2024-08-05	2026-06-30	1.3.0.0
1.2.1.1	1.2	1.1	aktif	2024-03-11	2026-06-30	1.3.0.0
1.2.1.0	1.2	1.0	aktif	2023-03-08	2026-06-30	1.3.0.0
1.2.0.2	1.2	0,2	aktif	2022-11-16	2026-06-30	1.3.0.0
1.2.0.1	1.2	0,1	aktif	2022-10-26	2026-06-30	1.3.0.0
1.2.0.0	1.2	0.0	aktif	2022-07-21	2026-06-30	1.3.0.0
1.1.1.0	1.1	1.0	aktif	2022-04-19	2026-06-30	1.2.1.0
1.1.0.0	1.1	0.0	usang	2021-11-19	2025-03-15	1.1.1.0
1.0.5.1	1.0	5.1	usang	2021-10-01	2023-01-30	1.1.0.0
1.0.5.0	1.0	5.0	usang	2021-07-27	2023-01-30	1.1.0.0
1.0.4.2	1.0	4.2	usang	2021-06-01	2023-01-30	1.1.0.0
1.0.4.1	1.0	4.1	usang	2020-12-08	2023-01-30	1.1.0.0
1.0.4.0	1.0	4.0	usang	2020-10-12	2023-01-30	1.1.0.0
1.0.3.0	1.0	3.0	usang	2020-08-03	2023-01-30	1.1.0.0
1.0.2.2	1.0	2.2	usang	2020-03-09	2022-07-29	1.0.3.0
1.0.2.1	1.0	2.1	usang	2019-11-22	2022-07-29	1.0.3.0
1.0.2.0	1.0	2.0	usang	2019-11-08	2020-05-19	1.0.3.0
1.0.1.2	1.0	1.2	usang	2019-10-15	—	—
1.0.1.1	1.0	1.1	usang	2019-08-13	—	—

Versi	Versi besar	Versi minor	Status	Dirilis	Akhir hidup	Tingkatkan ke:
1.0.1.0. *	1.0	1.0. *	usang	2019-07-02 dan sebelumnya	—	—

end-of-lifePerencanaan versi mesin utama

Versi mesin Neptunus hampir selalu mencapai akhir masa pakainya pada akhir seperempat kalender. Pengecualian hanya terjadi ketika masalah keamanan atau ketersediaan penting muncul.

Ketika versi mesin mencapai akhir masa pakainya, Anda akan diminta untuk meningkatkan basis data Neptunus Anda ke versi yang lebih baru.

Secara umum, versi mesin Neptunus terus tersedia sebagai berikut:

- Versi mesin minor: Versi mesin minor tetap tersedia setidaknya selama 6 bulan setelah dirilis.
- Versi mesin utama: Versi mesin utama tetap tersedia setidaknya selama 12 bulan setelah dirilis.

Setidaknya 3 bulan sebelum versi mesin mencapai akhir masa pakainya, AWS akan mengirimkan pemberitahuan email otomatis ke alamat email yang terkait dengan AWS akun Anda dan memposting pesan yang sama ke Dasbor AWS Kesehatan Anda. Ini akan memberi Anda waktu untuk merencanakan dan bersiap untuk meningkatkan.

Ketika versi mesin mencapai akhir masa pakainya, Anda tidak akan lagi dapat membuat cluster atau instance baru menggunakan versi itu, juga tidak akan dapat membuat instance menggunakan versi itu.

Versi mesin yang benar-benar mencapai akhir masa pakainya akan secara otomatis ditingkatkan selama jendela pemeliharaan. Pesan yang dikirimkan kepada Anda 3 bulan sebelum akhir masa pakai versi mesin akan berisi detail tentang apa yang akan melibatkan pembaruan otomatis ini, termasuk versi yang akan Anda upgrade secara otomatis, dampaknya pada cluster DB Anda, dan tindakan yang kami rekomendasikan.

> **Important**
>
> Anda bertanggung jawab untuk menjaga versi mesin database Anda tetap terkini. AWS mendesak semua pelanggan untuk meningkatkan basis data mereka ke versi mesin terbaru untuk mendapatkan keuntungan dari perlindungan keamanan, privasi, dan ketersediaan terkini. Jika Anda mengoperasikan database Anda pada mesin atau perangkat lunak yang tidak didukung setelah tanggal penghentian ("Legacy Engine"), Anda menghadapi kemungkinan risiko keamanan, privasi, dan operasional yang lebih besar, termasuk peristiwa downtime.
>
> Pengoperasian database Anda pada mesin apa pun tunduk pada Perjanjian yang mengatur penggunaan AWS Layanan oleh Anda. Mesin Legacy umumnya tidak tersedia. AWS tidak lagi memberikan dukungan untuk Legacy Engine, dan AWS dapat membatasi akses atau penggunaan Legacy Engine kapan saja, jika AWS menentukan Legacy Engine menimbulkan risiko keamanan atau kewajiban, atau risiko bahaya, terhadap Layanan, Afiliasinya AWS, atau pihak ketiga mana pun. Keputusan Anda untuk terus menjalankan Konten Anda di Legacy Engine dapat mengakibatkan Konten Anda menjadi tidak tersedia, rusak, atau tidak dapat dipulihkan. Database yang berjalan pada Legacy Engine tunduk pada Pengecualian Service Level Agreement (SLA).
>
> DATABASE DAN PERANGKAT LUNAK TERKAIT YANG BERJALAN PADA MESIN LAMA MENGANDUNG BUG, KESALAHAN, CACAT, KOMPONEN AND/OR BERBAHAYA. DENGAN DEMIKIAN, DAN TERLEPAS DARI APA PUN YANG BERTENTANGAN DALAM PERJANJIAN ATAU KETENTUAN LAYANAN, AWS MENYEDIAKAN MESIN LAMA "SEBAGAIMANA ADANYA."

Mesin Amazon Neptunus versi 1.4.7.0 (2026-03-03)

Pada 2026-03-03, engine versi 1.4.7.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Important**
>
> Upgrade Sementara Tidak Tersedia
>
> Upgrade ke mesin Neptunus versi 1.4.7.0 tidak tersedia saat ini karena masalah yang diketahui terkait dengan peningkatan cluster Global Database (GDB). Kami berharap dapat mengaktifkan peningkatan dalam beberapa minggu.

Fitur Baru dalam Rilis Mesin Ini

- OpenCypher membaca dari dukungan S3 untuk file Parket dan CSV melalui OC. Lihat the section called “neptune.read ()” dokumentasinya.
- Fungsi kueri geospasial OpenCypher. Rilis ini mencakup 12 fungsi Tipe Spasial berdasarkan standar ISO/IEC 13249-3:2016, tipe properti Geometri baru untuk POINT yang disimpan dalam indeks geopetrik baru untuk pengambilan cepat, dan dukungan untuk format Teks Terkenal (WKT). Lihat Data Spasial dan Fungsi Spasial dokumentasinya.

Perbaikan dalam Rilis Mesin Ini

- Peningkatan kinerja kueri untuk subkueri SPARQL yang mengembalikan set hasil kecil, termasuk yang memiliki nilai LIMIT kecil
- Peningkatan kinerja kueri dalam kasus di mana variabel dibatasi oleh sejumlah besar nilai konstan (misalnya, oleh klausa NILAI SPARQL, atau klausa UNWIND) OpenCypher
- Perbaikan untuk kueri penyisipan latensi rendah melalui beberapa pengoptimalan ke sisipan kamus
- Menambahkan langkah bahasa Gremlin baru ke mesin DFE (lihat cakupan langkah Gremlin di DFE).
  - Langkah-langkah jalur dan traversal: `order(local)`
  - Langkah-langkah agregat dan pengumpulan: `dedup(local)`
- Peningkatan kinerja untuk OpenCypher kueri termasuk`COLLECT(DISTINCT ...)`. Penulisan ulang yang dijelaskan dalam kueri Rewriting COLLECT (DISTINCT...) tidak lagi diperlukan saat menggunakan engine versi 1.4.7.0 atau yang lebih baru.

Perbaikan Cacat dalam Rilis Mesin Ini

Perbaikan umum:

- Perbaiki untuk beban massal menjadi tidak responsif saat memuat sejumlah besar file tepi
- Memperbaiki masalah patch cluster database global yang memengaruhi pembaruan klaster sekunder dari rilis 1.4.0.0, 1.4.1.0, dan 1.4.2.0.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan kluster DB ke versi 1.4.7.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.4.7.0

Anda dapat memutakhirkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.7.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.7.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri,

serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus versi 1.4.6.3 (2025-12-18)

Pada 2025-12-18, engine versi 1.4.6.3 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Cacat diperbaiki dalam rilis mesin ini (1.4.6.3)

Perbaikan umum

- Beberapa pengecualian yang keliru diklasifikasikan sebagai Kesalahan Server Internal sekarang dilaporkan dengan benar sebagai Pengecualian Keluar dari Memori selama eksekusi Data Flow Engine (DFE).
- Memperbaiki masalah yang dapat menyebabkan instance Database Neptunus gagal selama startup saat validasi ID tepi sebaris mengalami jenis entri kamus yang tidak terduga. Validasi ID

tepi sebaris sekarang hanya berjalan untuk konfigurasi mesin kueri Gremlin dan menangani entri kamus non-URI dengan anggun.
- Memperbaiki masalah saat pemuat massal Neptunus gagal terhubung ke S3 di wilayah tertentu karena resolusi titik akhir yang salah.

Perbaikan OpenCypher

- Memperbaiki kerusakan mesin yang terkadang terjadi dalam kueri menggunakan`CALL`.
- Batas waktu tetap dan penanganan pembatalan untuk kueri mutasi.
- Memperbaiki bug di mana `MERGE` klausa memberikan hasil yang salah ketika salah satu nilai properti diteruskan ke `MERGE` memiliki nilai null.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.4.6.3, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.4.6.3

Anda dapat meningkatkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
```

```
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.6.3 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.6.3 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus versi 1.4.6.2 (2025-11-18)

Pada 2025-11-18, versi mesin 1.4.6.2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Cacat diperbaiki dalam rilis mesin ini

Perbaikan umum

- Beberapa pengecualian yang secara keliru diklasifikasikan sebagai Pengecualian Kegagalan Internal (IFEs) sekarang dilaporkan dengan benar sebagai Pengecualian Modifikasi Bersamaan (CMEs) selama eksekusi Data Flow Engine (DFE).

Perbaikan Gremlin

- Meningkatkan stabilitas mesin untuk Gremlin di atas DFE

Perbaikan OpenCypher

- Memperbaiki masalah yang diperkenalkan dalam rilis mesin Neptunus 1.4.6.0 di mana kueri OpenCypher MERGE salah menangani arah hubungan masuk (←). Kueri menggunakan pola seperti MERGE (n) ← [:type] - (m) sekarang membuat hubungan dengan arah yang benar.
- Memperbaiki operasi komit yang gagal pada transaksi penulis yang tidak diputar kembali dengan benar.
- Paket kueri yang dioptimalkan sekarang didukung untuk kueri GABUNGAN yang mereferensikan nilai parameter.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.4.6.2, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.4.6.2

Anda dapat meningkatkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.6.2 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.6.2 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWSPremium Support.

Mesin Amazon Neptunus versi 1.4.6.1 (2025-09-18)

Pada 2025-09-18, engine versi 1.4.6.1 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Cacat diperbaiki dalam rilis mesin ini

Perbaikan umum

- Menghapus pemeriksaan jaringan untuk cluster yang menggunakan non-RFC 1918 sesuai rentang IP pribadi yang diperkenalkan di Release. Mesin Amazon Neptunus versi 1.4.6.0 (2025-09-02)

Perbaikan Gremlin

- Memperbaiki masalah manajemen koneksi websocket dengan transaksi.
- Memperbaiki masalah restart instance yang jarang terjadi saat menggunakan mode DFE gremlin

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan kluster DB ke versi 1.4.6.1, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.4.6.1

Anda dapat memutakhirkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.6.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.6.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus versi 1.4.6.0 (2025-09-02)

Pada 2025-09-02, engine versi 1.4.6.0 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Warning

Versi mesin 1.4.6.0 mencakup pemeriksaan jaringan baru untuk cluster yang menggunakan non-RFC 1918 yang sesuai dengan rentang IP pribadi untuk VPC basis data tanpa otentikasi IAM. Jika Anda memiliki konfigurasi VPC dan IAM ini, Anda perlu memperbarui VPC database Anda untuk menggunakan rentang IP pribadi RFC 1918 yang and/or mengaktifkan otentikasi IAM untuk menghindari kesalahan dengan kueri setelah memutakhirkan ke 1.4.6.0.

Fitur baru dalam rilis mesin ini

- Connect ke Neptunus menggunakan endpoint publik. Untuk informasi lebih lanjut, lihat titik akhir publik Neptunus.

Perbaikan dalam rilis mesin ini

Perbaikan Umum

- Peningkatan kinerja SPARQL untuk operasi pembaruan.
- Peningkatan OpenCypher kinerja untuk`CREATE`,`MERGE`, dan `SET` (mutasi) operasi.
- Peningkatan OpenCypher kinerja untuk operasi CALL Subquery.

Perbaikan OpenCypher

- Menambahkan petunjuk kueri baru untuk mendukung batas waktu tingkat kueri.

Cacat diperbaiki dalam rilis mesin ini

Perbaikan Gremlin

- Koneksi ke sesi Gremlin harus terjadi pada saluran yang sama yang membuatnya, artinya tidak mungkin menghubungkan beberapa instance klien ke sesi yang sama.
- Sesi Gremlin selalu ditutup ketika klien ditutup, tetapi mereka sekarang juga akan menutup untuk server yang memulai penutupan koneksi yang mencegah koneksi ulang yang tidak diinginkan atau tidak terduga.
- Kebocoran memori tetap untuk kueri Gremlin yang membaca data tipe gumpalan besar.

Perbaikan OpenCypher

- Penanganan variabel tetap setelah `CALL` subquery.
- Memperbaiki masalah dengan `reduce` fungsi untuk menangani skenario luapan aritmatika dengan benar.
- Memperbaiki kebocoran memori yang memengaruhi kueri parameter saat Cache Paket Kueri diaktifkan.
- Memperbaiki masalah dengan `NOT EXISTS` digunakan dalam `WHERE` klausa kompleks.
- Perbaiki untuk Cuncurrent Memory Exception (CMEs) yang salah dilaporkan sebagai. BadRequestException

Perbaikan SPARQL

- Memperbaiki pesan kesalahan untuk SPARQL `LOAD/UNLOAD` saat sumber jarak jauh tidak tersedia.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.4.6.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.4.6.0

Anda dapat memutakhirkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.6.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.6.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWSPremium Support.

Mesin Amazon Neptunus versi 1.4.5.1 (2025-06-30)

Pada 2025-06-30, engine versi 1.4.5.1 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Cacat diperbaiki dalam rilis mesin ini

Perbaikan umum

- Memperbaiki masalah dengan penskalaan tanpa server yang memengaruhi semua bahasa kueri.
- Memperbaiki masalah saat kesalahan internal dikembalikan pada beberapa OpenCypher kueri saat berjalan `collect(distinct())` di daftar.
- Memperbaiki masalah saat kueri FTS bersamaan menyebabkan kueri FTS bersamaan lainnya dihentikan saat menggunakan tanpa server. OpenSearch

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.4.5.1, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.4.5.1

Anda dapat meningkatkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.5.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.5.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri,

serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus versi 1.4.5.0 (2025-04-09)

Pada 2025-04-09, engine versi 1.4.5.0 secara umum digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Warning**
>
> Kami telah menghentikan sementara peningkatan ke 1.4.5.0 karena masalah yang dapat terjadi dalam konfigurasi Tanpa Server tertentu. Kami merekomendasikan untuk meningkatkan ke versi mesin 1.4.5.1. Upgrade ke 1.4.5.0 telah dinonaktifkan sementara.

Fitur baru dalam rilis mesin ini

- Menambahkan langkah bahasa Gremlin baru ke mesin DFE.

- Langkah-langkah jalur dan traversal: asDate (), dateAdd (), dateDiff (), gagal (), Inject (), label (), path (), project (), repeat (), sack (), select (), unfold (), disjunct (), drop (), identity (), intersect (), length (), loop (), barrier (), order (), range (), reverse (), sample (), cap (), split (), filter (), flatMap (), map (), sideEffect (), union (), index ()
- Langkah agregat dan pengumpulan: agregat (global), gabungkan (), hitung (), dedup (global), lipat (), grup (), groupCount ()
- Langkah-langkah matematika: max (), mean (), min (), sum ()
- Langkah-langkah elemen: otherV (), elementMap (), element (), V (), out (), in (), both (), oute (), inE (), boThe (), outV (), inV (), BothV (), otherV ()
- Langkah-langkah properti: properti (), key (), valueMap (), value ()
- Langkah-langkah filter: and (), coalesce (), coin (), is (), local (), none (), not (), or (), where ()
- Langkah-langkah manipulasi string: concat (), lTrim (), rTrim (), substring (), toLower (), toupper (), trim ()
- Predikat:
  - Bandingkan: eq, neq, lt, lte, gt, gte
  - Berisi: di dalam, tanpa
  - TextP: EndingWith, berisi,,, notContains notStartingWith notEndingWith
  - P: dan, atau, antara, di luar, di dalam

Untuk detail tentang semua langkah Gremlin yang tersedia di DFE, lihat. Cakupan langkah Gremlin di DFE

Perbaikan dalam rilis mesin ini

Perbaikan Umum

- Peningkatan waktu tunggu kunci log kueri lambat. Log kueri lambat sekarang menyertakan metrik waktu tunggu untuk kunci bersama dan eksklusif. Mereka disimpan sebagai bagian dari setiap transaksi jika terjadi promosi baca-tulis yang malas. Metrik ini muncul di bagian StorageCounters dari log kueri lambat.
- Menjatuhkan dukungan untuk rangkaian sandi berikut:
  - TLS_ECDHE_RSA_WITH_AES_128_CBC_ SHA256
  - TLS_ECDHE_RSA_WITH_AES_256_CBC_ SHA384
  - TLS_ECDHE_ECDSA_DENGAN_AES_128_CBC_ SHA256

- TLS_ECDHE_ECDSA_DENGAN_AES_256_CBC_ SHA384

Perbaikan Gremlin

- Menambahkan banyak langkah baru ke bahasa Gremlin. Untuk informasi selengkapnya, lihat Cakupan langkah Gremlin di DFE.

Peningkatan OpenCypher

- CREATE, MERGE, dan SET (mutasi) peningkatan kinerja.
- CALL Peningkatan kinerja Subquery.
- Dukungan untuk dukungan header trailing HTTP untuk respons OpenCypher multi-bagian. Untuk informasi selengkapnya, lihat Header tambahan HTTP opsional.
- Menambahkan fungsi temporal hari, bulan, dan tahun ke OpenCypher. Untuk informasi lebih lanjut, lihat Fungsi temporal.

```
RETURN day(datetime('2021-06-03T01:48:14Z'))
{
  "results": [{
      "day(datetime('2021-06-03T01:48:14Z'))": 3
    }]
}
```

Cacat diperbaiki dalam rilis mesin ini

Perbaikan umum

- Memperbaiki masalah yang menjatuhkan file Audit/SlowQueryLog log.

Perbaikan Gremlin

- Memperbaiki masalah dengan kueri Gremlin yang berjalan dengan fitur Result Cache dinonaktifkan. Kueri yang diakhiri dengan iterate () mengembalikan hasil alih-alih mengembalikan respons kosong.

- Memperbaiki masalah dengan Cache Hasil Gremlin yang disebabkan oleh kueri bersamaan dengan kunci yang sama. Salah satu kueri yang berjalan bersamaan salah mengembalikan hasil alih-alih mengembalikan hasil kosong.
- Memperbaiki masalah dengan kueri ekspor Amazon S3 yang akan menyebabkan kueri gagal dalam unggahan multi-bagian Amazon S3 karena batas waktu atau pembatalan, dengan meningkatkan waktu pembersihan.
- Memperbaiki masalah izin yang terkait dengan ekspor Amazon S3 Gremlin.

Perbaikan SPARQL

- Memperbaiki masalah dalam penanganan kueri SPARQL yang mendeklarasikan beberapa basis yang hanya akan menyebabkan deklarasi awal IRIs yang akan digunakan.
- Memperbaiki masalah dalam penanganan `REPLACE` fungsi SPARQL menggunakan string pola tidak valid yang akan menyebabkan kesalahan dikembalikan.
- Memperbaiki masalah dalam penanganan `REPLACE` fungsi SPARQL menggunakan flag case-insensitivity (`"i"`) dengan data unicode.
- Memperbaiki masalah dalam penguraian kueri SPARQL menggunakan urutan escape yang tidak valid `\u` dan `\U` codepoint yang dapat menyebabkan tidak ada respons yang dikembalikan.
- Memperbaiki masalah dalam `IRI` fungsi SPARQL yang tidak selalu benar menyelesaikan relatif IRIs terhadap basis IRI saat ini.
- Memperbaiki masalah yang menyebabkan `SPARQL INSERT DATA` dan `DELETE DATA` memperbarui menggunakan nama awalan untuk gagal menyelesaikan dengan benar relatif IRIs terhadap basis IRI saat ini.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan kluster DB ke versi 1.4.5.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.4.5.0

Anda dapat meningkatkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.5.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.5.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
    proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWSPremium Support.

Mesin Amazon Neptunus versi 1.4.4.0 (2025-02-24)

Pada 2025-02-24, engine versi 1.4.4.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Warning

Cache rencana kueri sementara tidak didukung untuk kasus penggunaan mengeksekusi kueri berparameter yang melibatkan nilai parameter numerik, karena bug dalam menangani penggunaan duplikat parameter tipe numerik dalam kueri. Misalnya:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
      UNION
      MATCH (n:show) WHERE n.duration>=$minutes RETURN n

      parameters={"minutes":130}
```

Kueri yang melakukan banyak pencarian indeks pada pernyataan atau indeks kamus dapat melihat regresi kinerja 5%. Misalnya - mendapatkan hitungan semua simpul, atau mendapatkan semua simpul tidak akan terpengaruh. `id` Mendapatkan semua properti dari semua simpul bisa melihat regresi hingga 5%.

Cacat diperbaiki dalam rilis mesin ini

Perbaikan umum

- Memperbaiki masalah `WITH` ketika klausa yang berisi tanda bintang (`*`) dan ekspresi aliasing menyebabkan penguraian kueri yang salah di OpenCypher.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.4.4.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.4.4.0

Anda dapat meningkatkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.4.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.4.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri,

serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus versi 1.4.3.0 (2025-01-21)

Pada 2025-01-21, engine versi 1.4.3.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Warning**
>
> Cache paket kueri sementara tidak didukung untuk kasus penggunaan mengeksekusi kueri berparameter yang melibatkan nilai parameter numerik, karena bug dalam menangani penggunaan duplikat parameter tipe numerik dalam kueri. Misalnya:
>
> ```
> MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
>       UNION
>       MATCH (n:show) WHERE n.duration>=$minutes RETURN n
> ```

```
      parameters={"minutes":130}
```

Kueri yang melakukan banyak pencarian indeks pada pernyataan atau indeks kamus dapat melihat regresi kinerja 5%. Misalnya - mendapatkan hitungan semua simpul, atau mendapatkan semua simpul tidak akan terpengaruh. `id` Mendapatkan semua properti dari semua simpul bisa melihat regresi hingga 5%.

Fitur baru dalam rilis mesin ini

- Mengekspor hasil kueri Gremlin ke Amazon S3. Mengekspor hasil kueri Gremlin langsung ke Amazon S3. Fitur ini memungkinkan Anda menangani hasil kueri besar secara efisien dengan mengekspornya ke bucket Amazon S3, alih-alih mengembalikannya sebagai respons kueri.

```
g.V().
    hasLabel('Comment').
    valueMap().
    call('neptune.query.exportToS3', [
    'destination': 's3://your-bucket/path/result.json',
    'format': 'GraphSONv3',
    'keyArn': 'optional-kms-key-arn'
  ])
```

- Contoh R7i. Keluarga instans R7i, hingga 48xlarge, sekarang tersedia di wilayah berikut:
  - ap-northeast-1 - Asia Pasifik (Tokyo)
  - ap-northeast-2 - Asia Pasifik (Seoul)
  - ap-south-1 - Asia Pasifik (Mumbai)
  - ap-southeast-1 - Asia Pasifik (Singapura)
  - ap-southeast-2 - Asia Pasifik (Sydney)
  - ap-southeast-3 - Asia Pasifik (Jakarta)
  - ca-central-1 - Kanada (Tengah)
  - eu-central-1 - Eropa (Frankfurt)
  - eu-north-1 - Eropa (Stockholm)
  - eu-south-2 - Eropa (Spanyol)
  - eu-west-1 - Eropa (Irlandia)
  - eu-west-2 - Eropa (London)

- eu-west-3 - Eropa (Paris)
- us-east-1 - AS Timur (Virginia Utara)
- us-east-2 - AS Timur (Ohio)
- us-west-1 - AS Barat (California Utara)
- us-west-2 - AS Barat (Oregon)

Perbaikan dalam rilis mesin ini

Perbaikan Umum

- Dukungan mode lab untuk pengumpulan sampah kamus (GC).

  Saat diaktifkan, entri kamus yang tidak digunakan dibersihkan oleh pekerjaan latar belakang. Itu tidak mengurangi`VolumeBytesUsed`, itu membebaskan ruang dalam indeks untuk sisipan baru. Tingkat pertumbuhan cenderung `VolumeBytesUsed` lebih kecil ketika kamus GC diaktifkan relatif terhadap saat tidak. Ini berfungsi untuk data grafik properti (disisipkan melalui Gremlin atau OpenCypher) ketika parameter tidak diaktifkan. `neptune_streams` Untuk informasi selengkapnya, lihat Pengumpulan sampah kamus Neptunus

Cacat diperbaiki dalam rilis mesin ini

Perbaikan umum

- Memperbaiki dua masalah kebocoran memori yang memengaruhi FreeableMemory saat mesin DFE digunakan.

Perbaikan OpenCypher

- Selesaikan masalah dengan MERGE ON MATCH/ON CREATE untuk baris duplikat.

```
UNWIND [1, 1] AS id
MERGE (n:Person {id: id})
  ON CREATE SET n.p = 5
  ON MATCH SET n.p = 6
```

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.4.3.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.4.3.0

Anda dapat memutakhirkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.3.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.3.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri,

serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus versi 1.4.2.0 (2024-12-19)

Pada 2024-12-19, engine versi 1.4.2.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Warning**
>
> Cache rencana kueri sementara tidak didukung untuk kasus penggunaan mengeksekusi kueri berparameter yang melibatkan nilai parameter numerik, karena bug dalam menangani penggunaan duplikat parameter tipe numerik dalam kueri. Misalnya:
>
> ```
> MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
>       UNION
>       MATCH (n:show) WHERE n.duration>=$minutes RETURN n
> ```

```
      parameters={"minutes":130}
```

Kueri yang melakukan banyak pencarian indeks pada pernyataan atau indeks kamus dapat melihat regresi kinerja 5%. Misalnya - mendapatkan hitungan semua simpul, atau mendapatkan semua simpul tidak akan terpengaruh. `id` Mendapatkan semua properti dari semua simpul bisa melihat regresi hingga 5%.

Cacat diperbaiki dalam rilis mesin ini

Perbaikan umum

- Memperbaiki rencana eksekusi dan kinerja untuk kueri dengan beberapa akses properti di atas variabel yang sama di dalam PENCOCOKAN OPSIONAL dan pemahaman daftar. Misalnya:

```
MATCH (n)
  WHERE n.name = 'A'
OPTIONAL MATCH (n)-[:knows]->(m)
  WHERE m.name = 'C' or m.city = 'B'
RETURN m
```

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.4.2.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.4.2.0

Anda dapat memutakhirkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.2.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.2.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
    proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus versi 1.4.1.0 (2024-11-21)

Pada 2024-11-21, engine versi 1.4.1.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Warning

Cache rencana kueri sementara tidak didukung untuk kasus penggunaan mengeksekusi kueri berparameter yang melibatkan nilai parameter numerik, karena bug dalam menangani penggunaan duplikat parameter tipe numerik dalam kueri. Contoh:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
      UNION
      MATCH (n:show) WHERE n.duration>=$minutes RETURN n

      parameters={"minutes":130}
```

Kueri yang melakukan banyak pencarian indeks pada pernyataan atau indeks kamus dapat melihat regresi kinerja 5%. Misalnya - mendapatkan hitungan semua simpul, atau mendapatkan semua simpul tidak akan terpengaruh. `id` Mendapatkan semua properti dari semua simpul bisa melihat regresi hingga 5%.

Fitur baru dalam rilis mesin ini

- Menambahkan dukungan untuk `CALL` subquery dengan subquery read-only, memungkinkan eksekusi operasi dalam lingkup yang ditentukan. `CALL`Subquery dijalankan sekali untuk setiap baris yang masuk dan variabel yang dikembalikan dalam subquery tersedia untuk lingkup luar dari

kueri terlampir. Variabel dari lingkup luar dapat diimpor ke `CALL` subquery menggunakan klausa impor. `WITH` Untuk informasi selengkapnya, lihat CALL dukungan subquery di Neptunus.

```
MATCH (origin:airport {code:"AUS"})-[:route]->(stopover)
CALL {
  WITH stopover
  MATCH (stopover)-[r:route]->(destination)
  RETURN destination
  ORDER BY r.dist DESC LIMIT 2
}
RETURN stopover, destination
```

- Ditambahkan fungsi OpenCypher. Kami memperkenalkan delapan fungsi baru untuk membantu string, operasi koleksi, dan penyortiran koleksi. Ini termasuk:`textIndexOf`,`collToSet`,`collSubtract`,`collIntersection`,`collSort`,`collSortMaps`,dan`collSortNodes`. Lihat fungsi Neptunus OpenCypher untuk deskripsi, parameter input, output, dan contoh.

Perbaikan dalam rilis mesin ini

Perbaikan Gremlin

- Parameter mode lab baru`AccurateQRCMemoryEstimation`. Cache hasil kueri Gremlin, ketika diaktifkan, memungkinkan caching hasil kueri pada database. Secara default perkiraan perkiraan digunakan untuk menentukan ukuran hasil cache. Dengan parameter mode lab ini `AccurateQRCMemoryEstimation` diaktifkan, estimasi ukuran untuk hasil cache akan menggunakan perkiraan ukuran yang akurat alih-alih perkiraan.
- Memperbaiki masalah dengan pengoptimalan filter “tidak” dalam kueri Gremlin yang dijalankan pada mesin eksekusi default. Masalah ini memengaruhi kueri saat tepi difilter menggunakan langkah not () yang dikombinasikan dengan salah satu langkah outV () /inV () /otherV (). Contoh kueri meliputi:
  - `g.E().hasLabel("knows").not(outV().hasId("5"))`
  - `g.V().has('airport','code','SDF').outE().where(not(otherV().has(id, within('1','5','7')))).count()`

Perbaikan OpenCypher

- Peningkatan kinerja untuk kueri yang menggunakan daftar statis besar atau peta. Kueri tertentu dengan UNWIND atas daftar besar peta bersarang yang digunakan untuk menyisipkan/meningkatkan node dengan properti melihat peningkatan kinerja yang signifikan.
- Memperkenalkan petunjuk kueri OpenCypher baru untuk menginstruksikan mesin untuk mengasumsikan tipe data yang konsisten untuk nilai yang digunakan dalam kueri. Lihat AssumeConsistentDataTypesuntuk detail tentang petunjuk kueri OpenCypher baru.
- Memperkenalkan satu set fungsi OpenCypher baru untuk menangani teks dan nilai koleksi.

Cacat diperbaiki dalam rilis mesin ini

Perbaikan Gremlin

- Memperbaiki masalah di jalur kode TinkerPop OSS untuk membangun representasi Bytecode dari kueri traversal ketika salah satu `withStrategies()/withoutStrategies()/with()` langkah digunakan pada objek `GraphTraversalSource` “g”. Masalah salah menambahkan instruksi baru ke Bytecode alih-alih mengganti instruksi yang ada untuk strategi yang sama dan menyebabkan ketidakcocokan kunci cache selama pembatalan cache hasil untuk menghapus hasil yang disimpan.

Perbaikan OpenCypher

- Perilaku yang dikoreksi untuk `` `~id`match `` dalam CREATE/MERGE/MATCH klausa. Saat menggunakan `` `~id` `` nilai yang tidak valid seperti tipe null atau non-string, pengecualian yang benar sekarang dilemparkan untuk CREATE/MERGE klausa dan hasil nol dikembalikan untuk klausa. `MATCH`
- <string>Memperbaiki IFE saat pengguna menggunakan nilai tipe yang tidak didukung dengan fungsi agregasi (yaitu sum ()).
- Memperbaiki masalah di mana beberapa kueri mutasi latensi rendah dari beban kerja kueri yang besar gagal dengan kesalahan. OutOfMemory

Perbaikan SPARQL

- Memperbaiki masalah log audit saat menangani kueri SPARQL yang berisi karakter. `'%'`

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.4.1.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.4.1.0

Anda dapat meningkatkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri,

serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWSPremium Support.

Mesin Amazon Neptunus versi 1.4.0.0 (2024-11-06)

Pada 2024-11-06, engine versi 1.4.0.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Engine release 1.3.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.3.0.0 ke engine versi 1.3.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.3` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, `neptune1.2` atau. dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.3.0.0 ke atas. Demikian pula, Anda harus menggunakan grup parameter cluster

1.4.0.0 untuk versi mesin 1.4.0.0 dan di atasnya. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

Warning

Cache rencana kueri sementara tidak didukung untuk kasus penggunaan mengeksekusi kueri berparameter yang melibatkan nilai parameter numerik, karena bug dalam menangani penggunaan duplikat parameter tipe numerik dalam kueri. Contoh:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
      UNION
      MATCH (n:show) WHERE n.duration>=$minutes RETURN n

      parameters={"minutes":130}
```

Kueri yang melakukan banyak pencarian indeks pada pernyataan atau indeks kamus dapat melihat regresi kinerja 5%. Misalnya - mendapatkan hitungan semua simpul, atau mendapatkan semua simpul tidak akan terpengaruh. `id` Mendapatkan semua properti dari semua simpul bisa melihat regresi hingga 5%.

Fitur baru dalam rilis mesin ini

- Ketika edge ditambahkan ke grafik properti tanpa ID eksplisit, secara default server menetapkan ID tepi berbasis UUID, yang disimpan dalam kamus. Sekarang dengan menetapkan parameter cluster baru,`neptune_enable_server_generated_edge_id = 1`, server akan menetapkan IDs menggunakan integer 8-byte yang dikelola secara internal, tanpa overhead kamus. Ini menghasilkan penghematan penyimpanan dan peningkatan kinerja kueri tanpa perubahan apa pun pada kueri. Fitur ini saat ini didukung untuk sisipan melalui bahasa kueri Gremlin saja.
- Menambahkan dukungan untuk eksekusi langkah limit () Gremlin di lintasan bersarang untuk mesin DFE.

```
g.V().project("foo").by(out().order().by(T.id).limit(1))
```

Perbaikan dalam rilis mesin ini

Perbaikan Umum

- Neptunus akan secara otomatis merebut kembali penyimpanan undo yang dimiliki oleh transaksi besar setelah transaksi selesai dan log tidak lagi diperlukan untuk pemulihan.
- Support untuk replika survival database global. Fitur ini memungkinkan cluster sekunder untuk tetap melayani permintaan baca selama instance penulis restart pada cluster utama. Sebelumnya, ketika instance penulis dimulai ulang, semua instance pembaca di cluster sekunder juga dimulai ulang. Dengan rilis ini, instance pembaca klaster sekunder terus menyajikan permintaan baca selama instance penulis dimulai ulang, meningkatkan ketersediaan baca di cluster.
- Log audit sekarang ditulis secara sinkron, yang menjamin bahwa setiap kueri dicatat. Ini dapat memengaruhi kinerja untuk kueri yang sangat besar (**>** 100kb) atau beban kerja throughput tinggi (**>** 1000 qps).

Perbaikan Gremlin

- Batas waktu per kueri secara default diberlakukan agar lebih kecil dari batas waktu tingkat cluster. Dalam rilis sebelumnya, pemeriksaan ini diperkenalkan tetapi perlu diaktifkan secara eksplisit melalui parameter mode lab ''. StrictTimeoutValidation Dengan rilis ini, 'StrictTimeoutValidation' akan diaktifkan secara default dan harus dinonaktifkan secara eksplisit untuk menjaga perilaku lama.

Perbaikan OpenCypher

- Dalam rilis sebelumnya, kami memperkenalkan dukungan format datetime yang diperluas, diaktifkan melalui parameter mode lab. `DatetimeMillisecond` Dukungan format datetime yang diperluas ini sekarang diaktifkan secara default.

Perbaikan SPARQL

- Tindakan IAM eksplisit baru untuk izin kueri.

```
Previously:
COPY: WriteDataViaQuery & ReadDataViaQuery
MOVE: WriteDataViaQuery & DeleteDataViaQuery
DELETEINSERT: ReadDataViaQuery & DeleteDataViaQuery
```

```
 Now,
COPY: WriteDataViaQuery & ReadDataViaQuery & DeleteDataViaQuery
MOVE: WriteDataViaQuery & ReadDataViaQuery & DeleteDataViaQuery
DELETEINSERT: ReadDataViaQuery, WriteDataViaQuery if there is INSERT clause,
 DeleteDataViaQuery if there is DELETE clause.
```

Cacat diperbaiki dalam rilis mesin ini

Perbaikan umum

- Memperbaiki masalah dengan instance Tanpa Server yang dapat menyebabkan restart basis data saat meningkatkan skala.
- Memperbaiki masalah yang terkait dengan manajemen file log audit yang dapat menyebabkan file log tidak dapat diakses untuk diunduh atau dirotasi, dan dalam beberapa kasus meningkatkan penggunaan CPU.
- Memperbaiki masalah kueri yang terkait dengan pengoptimalan yang menunda pembuatan keluaran peta di mesin DFE.
- Memperbaiki masalah yang menyebabkan stempel waktu yang tidak cocok antara log audit dan log kueri lambat.

Perbaikan Gremlin

- Menyelesaikan masalah dalam manajemen WebSocket koneksi Gremlin di mana kueri yang berjalan untuk waktu melebihi batas waktu idle koneksi dihentikan sebelum waktunya. Ini secara khusus mempengaruhi klien Python Gremlin menggunakan transportasi AIOHTTP.

Perbaikan OpenCypher

- Memperbaiki masalah pada langkah pengumpulan yang menyebabkan pengecualian kegagalan internal saat nilai nol hadir selama konstruksi kueri kumpulkan (berbeda (n)).
- Memperbaiki masalah yang `NullPointerException` mungkin terjadi dalam kueri saat cache paket kueri diaktifkan.
- Memperbaiki masalah yang mengevaluasi lebih banyak data daripada yang dibutuhkan saat kueri berisi klausa LIMIT.

- Memperbaiki masalah saat menggunakan operasi rentang (<, <=, >, >=) dalam kueri berparameter dengan cache rencana kueri akan menghasilkan hasil duplikat.
- Memperbaiki masalah yang mentransposisi kolom hasil saat operasi UNION dan UNION ALL dilakukan menggunakan koneksi baut.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.4.0.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.4.0.0

Anda dapat memutakhirkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWSPremium Support.

Mesin Amazon Neptunus versi 1.3.4.0 (2024-10-01)

Pada 2024-10-01, engine versi 1.3.4.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Engine release 1.3.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine

lebih awal dari 1.3.0.0 ke engine versi 1.3.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.3` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, `neptune1.2` atau. dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.3.0.0 ke atas. Demikian pula, Anda harus menggunakan grup parameter cluster 1.4.0.0 untuk versi mesin 1.4.0.0 dan di atasnya. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

Warning

Masalah terdeteksi dengan operasi pembaruan SPARQL 1.1 yang mungkin ada dalam kondisi tertentu ketika operator pembaruan digunakan dengan kebijakan otorisasi berbasis tindakan. Jika Anda menggunakan operasi pembaruan SPARQL 1.1 dengan kebijakan otorisasi berbasis tindakan, kami sarankan untuk meningkatkan ke versi mesin minor Neptunus terbaru (setidaknya 1.3.4.0) yang mencakup perbaikan untuk masalah ini. Cache paket kueri dinonaktifkan sementara untuk kueri berparameter yang melibatkan nilai parameter numerik karena masalah dalam menangani penggunaan duplikat parameter tipe numerik seperti pada kueri berikut:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$minutes RETURN n

parameters={"minutes":130}
```

Perbaikan dalam rilis mesin ini

- Menambahkan dukungan untuk eksekusi langkah limit () Gremlin di lintasan bersarang untuk mesin DFE.
- Menambahkan CloudWatch metrik yang terkait dengan cache hasil Gremlin seperti yang tercantum di bawah ini, yang dapat berguna dalam mendiagnosis dan menyetel latensi cache hasil. Lihat Metrik Neptunus untuk detailnya.

```
NumResultCacheHit
NumResultCacheMiss
```

```
ResultCacheSizeInBytes
ResultCacheItemCount
ResultCacheOldestItemTimestamp
ResultCacheNewestItemTimestamp
```

Cacat diperbaiki dalam rilis mesin ini

Perbaikan umum

- Memperbaiki masalah di mana dalam kasus yang jarang terjadi, mesin mogok alih-alih mengembalikan kesalahan kueri.

Perbaikan Gremlin

- Kami telah meningkatkan penanganan permintaan dan pelaporan kesalahan ketika klien atau proxy mengirim permintaan peningkatan soket web melalui koneksi established/used HTTP (sebelumnya 400 tanggapan dengan kesalahan “tidak ada skrip gremlin yang disediakan, kode MissingParameterException" dikembalikan).
- Mengoptimalkan penanganan `mergeV` langkah-langkah dengan pembaruan nilai properti kardinalitas tunggal. Misalnya, kueri di bawah ini sekarang didukung secara native di Neptunus.

```
g.mergeV([(T.id): 1234]). option(onMatch, ['age': single(20), 'name':
 single('alice'), 'city': set('miami')])
```

- Memperbaiki masalah evaluasi kueri DFE Gremlin yang menyebabkan kueri gagal dengan file. `InternalFailureException` Kesalahan ini terjadi dengan pola tertentu `select` seperti yang ditunjukkan pada contoh berikut:

```
g.V("1").as("a").as("b").select("a","b").dedup()
```

Perbaikan OpenCypher

- Memperbaiki masalah saat menjalankan `collect(distinct())` dengan nilai nol yang ada menyebabkan kesalahan dikembalikan.
- Memperbaiki masalah saat menjalankan kueri berparameter yang berisi filter rentang (**</<=/>/>=** terhadap nilai parameter) mengarah ke hasil. duplicate/missing

- Memperbaiki bug di mana mesin DFE menghasilkan lebih banyak output daripada yang diminta dalam kueri batas, yang dapat menyebabkan kesalahan memori.

Perbaikan SPARQL

- Memperbaiki masalah saat menjalankan kueri pembaruan SPARQL federasi pada kluster yang mendukung otentikasi IAM menyebabkan kesalahan dikembalikan.
- Memperbaiki izin berbasis tindakan untuk operasi pembaruan SPARQL 1.1.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.3.4.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.3.4.0

Anda dapat meningkatkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.4.0 \
    --allow-major-version-upgrade \
```

```
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.4.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus versi 1.3.3.0 (2024-08-05)

Pada 2024-08-05, engine versi 1.3.3.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Engine release 1.3.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.3.0.0 ke engine versi 1.3.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.3` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, `neptune1.2` atau. dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.3.0.0 ke atas. Demikian pula, Anda harus menggunakan grup parameter cluster 1.4.0.0 untuk versi mesin 1.4.0.0 dan di atasnya. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

> **Warning**
>
> Rilis mesin 1.3.3.0 memperkenalkan beberapa masalah potensial yang harus Anda waspadai. Lihat bagian di bawah ini Mengurangi masalah dalam rilis 1.3.3.0 untuk informasi lebih lanjut.

Cacat diperbaiki dalam rilis mesin ini

Perbaikan umum

- Memperbaiki masalah di mana mesin menjadi tidak stabil ketika ada sejumlah besar predikat dalam cache predikat.

Perbaikan OpenCypher

- Memperbaiki masalah di mana eksekusi kueri dapat tetap macet setelah pengecualian internal dilemparkan.

- Memperbaiki masalah di mana kueri dapat gagal dengan pengecualian internal saat menggunakan cache paket kueri.

Perbaikan SPARQL

- Memperbaiki masalah dengan SPARQL 1.1 Graph Store HTTP Protocol (GSP) yang mungkin ada dalam kondisi tertentu ketika GSP digunakan dengan kebijakan otorisasi berbasis tindakan.

Mengurangi masalah dalam rilis 1.3.3.0

- Kueri yang menggunakan nilai filter numerik dapat mengembalikan hasil yang salah saat menggunakan cache rencana kueri. Untuk menghindari masalah, gunakan petunjuk kueri `QUERY:PLANCACHE "disabled"` untuk melewati cache rencana kueri. Misalnya, gunakan:

```
USING QUERY:PLANCACHE "disabled"
MATCH (n:person)
WHERE n.yearOfBirth > $year
RETURN n

parameters={"year":1950}
```

- Kueri menggunakan nama parameter yang sama beberapa kali dapat gagal dengan kesalahan`Parameter name should not be a number and/or contain _internal_ or _modified_user_ string within it. These are reserved for planCache. Otherwise, rerun with HTTP parameter planCache=disabled`. Lewati cache rencana kueri seperti di atas dalam kasus seperti itu, atau duplikat parameter seperti dalam contoh ini:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$minutes RETURN n

parameters={"minutes":130}
```

  Gunakan petunjuk `QUERY:PLANCACHE "disabled"` atau modifikasi parameter:

```
MATCH (n:movie) WHERE n.runtime>=$rt_min RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$dur_min RETURN n
```

```
parameters={"rt_min":130, "dur_min":130}
```

- Kueri yang dijalankan dengan protokol Bolt dapat menghasilkan hasil yang salah jika kueri adalah kueri UNION atau UNION ALL. Untuk menghindari masalah ini, pertimbangkan untuk mengeksekusi kueri tertentu dengan titik akhir HTTP. Atau, jalankan setiap bagian dari serikat secara terpisah saat menggunakan protokol Bolt.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.3.3.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.3.3.0

Anda dapat memutakhirkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.3.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.3.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus versi 1.3.2.1 (2024-06-20)

Pada 2024-06-20, engine versi 1.3.2.1 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Engine release 1.3.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine

lebih awal dari 1.3.0.0 ke engine versi 1.3.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.3` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, `neptune1.2` atau. dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.3.0.0 ke atas. Demikian pula, Anda harus menggunakan grup parameter cluster 1.4.0.0 untuk versi mesin 1.4.0.0 dan di atasnya. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

Warning

Rilis mesin 1.3.2.1 memperkenalkan beberapa masalah potensial yang harus Anda waspadai. Lihat bagian di bawah ini Mengurangi masalah dalam rilis 1.3.2.1 untuk informasi lebih lanjut.

Cacat diperbaiki dalam rilis mesin ini

Perbaikan OpenCypher

- Bug terdeteksi dalam fitur cache rencana kueri untuk kueri berparameter yang berisi `WITH` klausa bagian dalam yang memiliki `SKIP` dan sebagai parameter. `LIMIT` SKIP/LIMIT Nilai tidak diparameterisasi dengan benar, dan sebagai hasilnya, eksekusi selanjutnya dari rencana kueri cache yang sama dengan nilai parameter yang berbeda masih akan mengembalikan hasil yang sama dengan eksekusi pertama. Ini telah diperbaiki.

```
# insert some nodes
UNWIND range(1, 10) as i CREATE (s {name: i}) RETURN s

# sample query
MATCH (p)
WITH p ORDER BY p.name SKIP $s LIMIT $l
RETURN p.name as res

# first time executing with {"s": 2, "l": 1}
{
  "results" : [ {
    "res" : 3
  } ]
```

```
}

# second time executing with {"s": 2, "l": 10}
# due to bug, produces
{
  "results" : [ {
    "res" : 3
  } ]
}
# with fix, produces correct results:
{
  "results" : [ {
    "res" : 3
  }, {
    "res" : 4
  }, {
    "res" : 5
  }, {
    "res" : 6
  }, {
    "res" : 7
  }, {
    "res" : 8
  }, {
    "res" : 9
  }, {
    "res" : 10
  } ]
}%
```

- Memperbaiki bug di mana kueri mutasi berparameter muncul `InternalFailureException` ketika parameter yang diteruskan belum ada dalam database.
- Memperbaiki bug di mana kueri Bolt berparameter macet setelah mencapai kondisi balapan selama pembersihan sumber daya kueri.

Perubahan 1.3.2.1 dibawa dari 1.3.2.0

Perbaikan dilakukan dari rilis mesin 1.3.2.0

Perbaikan umum

- Support untuk TLS versi 1.3 termasuk cipher suite TLS_AES_128_GCM_ dan SHA256 TLS_AES_256_GCM_. SHA384 TLS 1.3 adalah opsi - TLS 1.2 masih minimum.
- Dukungan tambahan OpenCypher untuk format dateime ada di lab_mode untuk versi ini. Kami mendorong Anda untuk mengujinya.

Perbaikan Gremlin

- TinkerPop Peningkatan 3.7.x
  - Memberikan perluasan besar bahasa Gremlin.
    - Langkah-langkah baru untuk memproses string, daftar, dan tanggal.
    - Sintaks baru untuk menentukan kardinalitas dalam langkah. `mergeV()`
    - `union()`sekarang dapat digunakan sebagai langkah awal.
    - Untuk mempelajari lebih lanjut tentang perubahan di 3.7.x, lihat dokumentasi TinkerPop pemutakhiran.
  - Saat memutakhirkan driver bahasa Gremlin klien untuk Java, perhatikan bahwa kelas serializer telah membatalkan beberapa penggantian nama. Anda perlu memperbarui paket dan penamaan kelas dalam file konfigurasi Anda dan dalam kode, jika ditentukan.
- `StrictTimeoutValidation`(hanya jika diaktifkan melalui labmode `StrictTimeoutValidation` dengan menyertakan`StrictTimeoutValidation=enabled`): Ketika `StrictTimeoutValidation` parameter memiliki nilai`enabled`, nilai batas waktu per kueri yang ditentukan sebagai opsi permintaan atau petunjuk kueri tidak dapat melebihi nilai yang ditetapkan secara global dalam grup parameter. Dalam kasus seperti itu, Neptunus akan melempar a. `InvalidParameterException` Pengaturan ini dapat dikonfirmasi sebagai respons pada `/status` titik akhir saat nilainya`disabled`, dan di Neptunus versi 1.3.2.0 dan 1.3.2.1 nilai default parameter ini adalah. `Disabled`

Perbaikan OpenCypher

- Mesin Amazon Neptune versi 1.3.2.0 memberikan throughput hingga 9x lebih cepat dan 10x lebih tinggi untuk kinerja kueri OpenCypher vs rilis mesin sebelumnya.

- Kueri latensi rendah dan peningkatan kinerja throughput: Peningkatan kinerja keseluruhan untuk kueri OpenCypher latensi rendah. Versi baru juga meningkatkan throughput untuk kueri tersebut. Perbaikan lebih signifikan ketika kueri parameter digunakan.
- Support for Query Plan Cache: Ketika kueri dikirimkan ke Neptunus, string kueri diurai, dioptimalkan, dan diubah menjadi rencana kueri, yang kemudian dieksekusi oleh mesin. Aplikasi sering didukung oleh pola kueri umum yang dipakai dengan nilai yang berbeda. Cache rencana kueri dapat mengurangi latensi keseluruhan dengan menyimpan paket kueri dan dengan demikian menghindari penguraian dan pengoptimalan untuk pola berulang tersebut. Lihat Cache paket kueri di Amazon Neptunus untuk detail selengkapnya.
- Peningkatan Kinerja untuk kueri agregasi DISTINCT.
- Peningkatan kinerja untuk gabungan yang melibatkan variabel nullable.
- Peningkatan kinerja untuk kueri yang melibatkan tidak sama dengan predikat id (node/relasi).
- Dukungan diperpanjang untuk fungsionalitas datetime (Hanya diaktifkan melalui mode lab `DatetimeMillisecond` dengan menyertakan. `DatetimeMillisecond=enabled` Untuk informasi selengkapnya, lihat Dukungan sementara dalam implementasi Neptunus OpenCypher (Neptunus Analytics dan Neptunus Database 1.3.2.0 dan di atasnya).

Perbaikan cacat terbawa dari rilis mesin 1.3.2.0

Perbaikan umum

- Memperbarui pesan kesalahan Neptuneml saat memvalidasi akses ke bucket Graphlytics.

Perbaikan Gremlin

- Memperbaiki informasi label yang hilang dalam terjemahan kueri DFE, untuk skenario di mana langkah-langkah kontribusi non-jalur berisi label. Misalnya:

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'marko').
  has("name", TextP.regex("mark.*")).as("p1").
  not(out().has("name", P.within("peter"))).
  out().as('p2').
  dedup('p1', 'p2')
```

- Memperbaiki `NullPointerException` bug dalam terjemahan kueri DFE, yang terjadi ketika kueri dijalankan dalam dua fragmen DFE, dan fragmen pertama dioptimalkan ke node yang tidak memuaskan. Misalnya:

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'doesNotExists').
  has("name", TextP.regex("mark.*")).
  inject(1).
  V().
  out().
  has('name', 'vadas')
```

- Memperbaiki bug di mana Neptunus bisa melempar `InternalFailureException` ketika kueri ValueTraversal berisi di dalam by () modulator dan inputnya adalah Map. Misalnya:

```
g.V().
  hasLabel("person").
  project("age", "name").by("age").by("name").
  order().by("age")
```

Perbaikan OpenCypher

- Peningkatan operasi UNWIND (misalnya memperluas daftar nilai ke dalam nilai individual) untuk membantu mencegah situasi out of memory (OOM). Misalnya:

```
MATCH (n)-->(m)
WITH collect(m) AS list
UNWIND list AS m
RETURN m, list
```

- Optimalisasi id kustom tetap jika terjadi beberapa operasi MERGE di mana id disuntikkan melalui UNWIND. Misalnya:

```
UNWIND [{nid: 'nid1', mid: 'mid1'}, {nid: 'nid2', mid: 'mid2'}] as ids
MERGE (n:N {`~id`: ids.nid})
MERGE (m:M {`~id`: ids.mid})
```

- Memperbaiki ledakan memori saat merencanakan kueri kompleks dengan akses properti dan beberapa hop dengan hubungan dua arah. Misalnya:

```
MATCH (person1:person)-[:likes]->(res)-[:partOf]->(group)-[:knows]-(:entity {name:
 'foo'}),
      (person1)-[:knows]->(person2)-[:likes]-(res2), (comment)-[:presentIn]->(:Group
 {name: 'barGroup'}),
      (person1)-[:commented]->(comment2:comment)-[:partOf]->(post:Post), (comment2)-
[:presentIn]->(:Group {name: 'fooGroup'}),
      (comment)-[:contains]->(info:Details)-[:CommentType]->(:CommentType {name:
 'Positive'}),
      (comment2)-[:contains]->(info2:Details)-[:CommentType]->(:CommentType {name:
 'Positive'})
WHERE datetime('2020-01-01T00:00') <= person1.addedAfter <=
 datetime('2023-01-01T23:59') AND comment.approvedBy = comment2.approvedBy
MATCH (comment)-[:contains]->(info3:Details)-[:CommentType]->(:CommentType {name:
 'Neutral'})
RETURN person1, group.name, info1.value,  post.ranking, info3.value
```

- Kueri agregasi tetap dengan null sebagai kelompok berdasarkan variabel. Misalnya:

```
MATCH (n)
RETURN null AS group, sum(n.num) AS result
```

Perbaikan SPARQL

- Memperbaiki parser SPARQL untuk meningkatkan waktu parsing untuk kueri besar seperti INSERT DATA yang berisi banyak triple dan token besar.

Mengurangi masalah dalam rilis 1.3.2.1

- Kueri menggunakan nilai filter numerik dapat mengembalikan hasil yang salah saat menggunakan cache rencana kueri. Untuk menghindari masalah, gunakan petunjuk kueri `QUERY:PLANCACHE "disabled"` untuk melewati cache rencana kueri. Misalnya, gunakan:

```
USING QUERY:PLANCACHE "disabled"
MATCH (n:person)
WHERE n.yearOfBirth > $year
RETURN n

parameters={"year":1950}
```

- Kueri menggunakan nama parameter yang sama beberapa kali dapat gagal dengan kesalahan`Parameter name should not be a number and/or contain _internal_ or _modified_user_ string within it. These are reserved for planCache. Otherwise, rerun with HTTP parameter planCache=disabled`. Lewati cache rencana kueri seperti di atas dalam kasus seperti itu, atau duplikat parameter seperti dalam contoh ini:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$minutes RETURN n

parameters={"minutes":130}
```

  Gunakan petunjuk `QUERY:PLANCACHE "disabled"` atau modifikasi parameter:

```
MATCH (n:movie) WHERE n.runtime>=$rt_min RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$dur_min RETURN n

parameters={"rt_min":130, "dur_min":130}
```

- Kueri yang dijalankan dengan protokol Bolt dapat menghasilkan hasil yang salah jika kueri adalah kueri UNION atau UNION ALL. Untuk menghindari masalah ini, pertimbangkan untuk mengeksekusi kueri tertentu dengan titik akhir HTTP. Atau, jalankan setiap bagian dari serikat secara terpisah saat menggunakan protokol Bolt.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.3.2.1, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.3.2.1

Anda dapat memutakhirkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.2.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.2.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
    proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus versi 1.3.2.0 (2024-06-10)

Pada 2024-06-10, engine versi 1.3.2.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Note

Engine release 1.3.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.3.0.0 ke engine versi 1.3.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.3` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, `neptune1.2` atau. dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.3.0.0 ke atas. Demikian pula, Anda harus menggunakan grup parameter cluster 1.4.0.0 untuk versi mesin 1.4.0.0 dan di atasnya. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

Warning

Rilis mesin 1.3.2.0 memperkenalkan beberapa masalah potensial yang harus Anda waspadai. Lihat bagian di bawah Mengurangi masalah dalam rilis 1.3.2.0 untuk informasi lebih lanjut.

Perbaikan dalam rilis mesin ini

Perbaikan umum

- Support untuk TLS versi 1.3 termasuk cipher suite TLS_AES_128_GCM_ dan SHA256 TLS_AES_256_GCM_. SHA384 TLS 1.3 adalah opsi - TLS 1.2 masih minimum.

Perbaikan Gremlin

- TinkerPop Peningkatan 3.7.x
  - Memberikan perluasan besar bahasa Gremlin.
    - Langkah-langkah baru untuk memproses string, daftar, dan tanggal.
    - Sintaks baru untuk menentukan kardinalitas dengan langkahnya. `mergeV()`
    - `union()`Sekarang dapat digunakan sebagai langkah awal.
    - Untuk mempelajari lebih lanjut tentang perubahan di 3.7.x, lihat dokumentasi TinkerPop pemutakhiran.
  - Saat memutakhirkan driver bahasa Gremlin klien untuk Java, perhatikan bahwa kelas serializer telah membatalkan beberapa penggantian nama. Anda perlu memperbarui paket dan penamaan kelas dalam file konfigurasi Anda dan dalam kode, jika ditentukan.
- `StrictTimeoutValidation`(hanya jika diaktifkan melalui labmode `StrictTimeoutValidation` dengan menyertakan`StrictTimeoutValidation=enabled`): Ketika `StrictTimeoutValidation` parameter memiliki nilai`enabled`, nilai batas waktu per kueri yang ditentukan sebagai opsi permintaan atau petunjuk kueri tidak dapat melebihi nilai yang ditetapkan secara global dalam grup parameter. Dalam kasus seperti itu, Neptunus akan melempar a. `InvalidParameterException` Pengaturan ini dapat dikonfirmasi sebagai respons pada `/status` titik akhir saat nilainya`disabled`, dan di Neptunus versi 1.3.2.0 nilai default parameter ini adalah. `Disabled`

Peningkatan OpenCypher

- Mesin Amazon Neptune versi 1.3.2.0 memberikan throughput hingga 9x lebih cepat dan 10x lebih tinggi untuk kinerja kueri OpenCypher vs rilis mesin sebelumnya.
- Kueri latensi rendah dan peningkatan kinerja throughput: Peningkatan kinerja keseluruhan untuk kueri OpenCypher latensi rendah. Versi baru juga meningkatkan throughput untuk kueri tersebut. Perbaikan lebih signifikan ketika kueri parameter digunakan.

- Support for Query Plan Cache: Ketika kueri dikirimkan ke Neptunus, string kueri diuraikan, dioptimalkan, dan diubah menjadi rencana kueri, yang kemudian dieksekusi oleh mesin. Aplikasi sering didukung oleh pola kueri umum yang dipakai dengan nilai yang berbeda. Cache rencana kueri dapat mengurangi latensi keseluruhan dengan menyimpan paket kueri dan dengan demikian menghindari penguraian dan pengoptimalan untuk pola berulang tersebut.
- Peningkatan Kinerja untuk kueri agregasi DISTINCT.
- Peningkatan kinerja untuk gabungan yang melibatkan variabel nullable.
- Peningkatan kinerja untuk kueri yang melibatkan tidak sama dengan predikat id (node/relasi).
- Dukungan diperpanjang untuk fungsionalitas datetime (Hanya diaktifkan melalui mode lab `DatetimeMillisecond` dengan menyertakan. `DatetimeMillisecond=enabled` Untuk informasi selengkapnya, lihat Dukungan sementara dalam implementasi Neptunus OpenCypher (Neptunus Analytics dan Neptunus Database 1.3.2.0 dan di atasnya).

Cacat diperbaiki dalam rilis mesin ini

Perbaikan umum

- Memperbarui pesan kesalahan Neptuneml saat memvalidasi akses ke bucket Graphlytics.

Perbaikan Gremlin

- Memperbaiki informasi label yang hilang dalam terjemahan kueri DFE, untuk skenario di mana langkah-langkah kontribusi non-jalur berisi label. Misalnya:

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'marko').
  has("name", TextP.regex("mark.*")).as("p1").
  not(out().has("name", P.within("peter"))).
  out().as('p2').
  dedup('p1', 'p2')
```

- Memperbaiki `NullPointerException` bug dalam terjemahan kueri DFE, yang terjadi ketika kueri dijalankan dalam dua fragmen DFE, dan fragmen pertama dioptimalkan ke node yang tidak memuaskan. Misalnya:

```
g.withSideEffect('Neptune#useDFE', true).
  V().
```

```
  has('name', 'doesNotExists').
  has("name", TextP.regex("mark.*")).
  inject(1).
  V().
  out().
  has('name', 'vadas')
```

- Memperbaiki bug di mana Neptunus bisa melempar `InternalFailureException` ketika kueri ValueTraversal berisi di dalam by () modulator dan inputnya adalah Map. Misalnya:

```
g.V().
  hasLabel("person").
  project("age", "name").by("age").by("name").
  order().by("age")
```

Perbaikan OpenCypher

- Peningkatan operasi UNWIND (misalnya memperluas daftar nilai ke dalam nilai individual) untuk membantu mencegah situasi out of memory (OOM). Misalnya:

```
MATCH (n)-->(m)
WITH collect(m) AS list
UNWIND list AS m
RETURN m, list
```

- Optimalisasi id kustom tetap jika terjadi beberapa operasi MERGE di mana id disuntikkan melalui UNWIND. Misalnya:

```
UNWIND [{nid: 'nid1', mid: 'mid1'}, {nid: 'nid2', mid: 'mid2'}] as ids
MERGE (n:N {`~id`: ids.nid})
MERGE (m:M {`~id`: ids.mid})
```

- Memperbaiki ledakan memori saat merencanakan kueri kompleks dengan akses properti dan beberapa hop dengan hubungan dua arah. Misalnya:

```
MATCH (person1:person)-[:likes]->(res)-[:partOf]->(group)-[:knows]-(:entity {name:
 'foo'}),
      (person1)-[:knows]->(person2)-[:likes]-(res2), (comment)-[:presentIn]->(:Group
 {name: 'barGroup'}),
      (person1)-[:commented]->(comment2:comment)-[:partOf]->(post:Post), (comment2)-
[:presentIn]->(:Group {name: 'fooGroup'}),
```

```
      (comment)-[:contains]->(info:Details)-[:CommentType]->(:CommentType {name:
 'Positive'}),
      (comment2)-[:contains]->(info2:Details)-[:CommentType]->(:CommentType {name:
 'Positive'})
WHERE datetime('2020-01-01T00:00') <= person1.addedAfter <=
 datetime('2023-01-01T23:59') AND comment.approvedBy = comment2.approvedBy
MATCH (comment)-[:contains]->(info3:Details)-[:CommentType]->(:CommentType {name:
 'Neutral'})
RETURN person1, group.name, info1.value,  post.ranking, info3.value
```

- Kueri agregasi tetap dengan null sebagai kelompok berdasarkan variabel. Misalnya:

```
MATCH (n)
RETURN null AS group, sum(n.num) AS result
```

Perbaikan SPARQL

- Memperbaiki parser SPARQL untuk meningkatkan waktu penguraian untuk kueri besar seperti INSERT DATA yang berisi banyak triple dan token besar.

Mengurangi masalah dalam rilis 1.3.2.0

- Untuk versi 1.3.2.0, kami telah mendeteksi masalah dalam cache rencana kueri saat `skip` atau digunakan dalam `WITH` klausa dalam dan `limit` diparameterisasi. Misalnya:

```
MATCH (n:Person)
WHERE n.age > $age
WITH n skip $skip LIMIT $limit
RETURN n.name, n.age

parameters={"age": 21, "skip": 2, "limit": 3}
```

  Dalam hal ini, nilai parameter untuk lewati dan batas dari rencana pertama akan diterapkan ke kueri berikutnya, juga, yang mengarah ke hasil yang tidak terduga.

  Mitigasi

Untuk mencegah masalah ini, tambahkan petunjuk kueri `QUERY:PLANCACHE "disabled"` saat mengirimkan kueri yang menyertakan sub-klausa batas lewati berparameter. and/or Atau, Anda dapat membuat hard-code nilai ke dalam kueri.

Opsi 1: Menggunakan Petunjuk Kueri untuk menonaktifkan cache paket:

```
Using QUERY:PLANCACHE "disabled"
MATCH (n:Person) WHERE n.age > $age
WITH n skip $skip LIMIT $limit
RETURN n.name, n.age

parameters={"age": 21, "skip": 2, "limit": 3}
```

Opsi 2: Menggunakan nilai hard-code untuk lewati dan batas:

```
MATCH (n:Person)
WHERE n.age > $age
WITH n skip 2 LIMIT 3
RETURN n.name, n.age

parameters={"age": 21}
```

- Kueri menggunakan nilai filter numerik dapat mengembalikan hasil yang salah saat menggunakan cache rencana kueri. Untuk menghindari masalah, gunakan petunjuk kueri `QUERY:PLANCACHE "disabled"` untuk melewati cache rencana kueri. Misalnya, gunakan:

```
USING QUERY:PLANCACHE "disabled"
MATCH (n:person)
WHERE n.yearOfBirth > $year
RETURN n

parameters={"year":1950}
```

- Kueri menggunakan nama parameter yang sama beberapa kali dapat gagal dengan kesalahan`Parameter name should not be a number and/or contain _internal_ or _modified_user_ string within it. These are reserved for planCache. Otherwise, rerun with HTTP parameter planCache=disabled`. Lewati cache rencana kueri seperti di atas dalam kasus seperti itu, atau duplikat parameter seperti dalam contoh ini:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
```

```
UNION
MATCH (n:show) WHERE n.duration>=$minutes RETURN n

parameters={"minutes":130}
```

Gunakan petunjuk `QUERY:PLANCACHE "disabled"` atau modifikasi parameter:

```
MATCH (n:movie) WHERE n.runtime>=$rt_min RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$dur_min RETURN n

parameters={"rt_min":130, "dur_min":130}
```

- Kueri yang dijalankan dengan protokol Bolt dapat menghasilkan hasil yang salah jika kueri adalah kueri UNION atau UNION ALL. Untuk menghindari masalah ini, pertimbangkan untuk mengeksekusi kueri tertentu dengan titik akhir HTTP. Atau, jalankan setiap bagian dari serikat secara terpisah saat menggunakan protokol Bolt.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan kluster DB ke versi 1.3.2.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.7.1`
- Gremlin versi terbaru didukung: `3.7.1`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.3.2.0

Anda dapat meningkatkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.2.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.2.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>  We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus versi 1.3.1.0 (2024-03-06)

Pada 2024-03-06, engine versi 1.3.1.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Engine release 1.3.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.3.0.0 ke engine versi 1.3.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.3` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, `neptune1.2` atau. dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.3.0.0 ke atas. Demikian pula, Anda harus menggunakan grup parameter cluster 1.4.0.0 untuk versi mesin 1.4.0.0 dan di atasnya. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

Perbaikan dalam rilis mesin ini

Perbaikan umum

- Neptunus telah meningkatkan peringatan yang ditunjukkan di profil/jelaskan.
- Menghapus kurva NIST EC usang dari grup bernama default yang digunakan selama negosiasi TLS. Kurva yang dihapus adalah sect409k1, sect409r1, dan sect571k1.

Perbaikan Gremlin

- Perhitungan statistik DFE yang ditingkatkan untuk menghindari instans Tanpa Server yang sangat tinggi NCUs .
- Peningkatan kinerja Gremlin untuk WITHIN.

Cacat diperbaiki dalam rilis mesin ini

Perbaikan Gremlin

- Perbaikan lain-lain pada rencana kueri Gremlin DFE.

- Perbaikan bug untuk kueri Gremlin dengan traversal opsional, misalnya, untuk kueri formulir `g.v () .hasLabel ('person') .group () .by (id ()) .by (__.in ('friend') .id () .fold ()) `, di mana tidak ada orang tanpa tepi teman yang dikelompokkan.
- Memperbaiki bug di mana kueri Gremlin yang berisi langkah-langkah gabungan di dalam by modulator menyebabkan kesalahan dikembalikan jika dijalankan menggunakan mesin DFE.
- Memperbaiki bug yang mencegah kueri hanya-baca yang berjalan di sesi Gremlin agar tidak berfungsi saat terhubung ke replika baca.
- Perbaikan bug di mana IAM ARN tidak ada dalam log audit untuk permintaan koneksi websocket awal yang berhasil untuk Gremlin.
- Menggabungkan langkah, mengidentifikasi cakupan langkah dengan DFE.
- Optimalisasi set karakteristik untuk seluruh paket DFE.

Perbaikan OpenCypher

- Perbaikan bug dalam klausa SET OpenCypher untuk memungkinkan pengaturan pada ekspresi non-variabel (yaitu: match (n: test) set (kasus ketika n.prop = 2 lalu n end) .prop = 3 return n.prop.
- Perbaikan bug untuk gagal kueri OpenCypher yang melibatkan agregasi dan urutan oleh.
- Peningkatan UNWIND dari daftar besar yang berisi peta statis.
- Perbaikan bug OpenCypher MERGE query menggunakan id kustom dengan nilai duplikat.

Perbaikan SPARQL

- Memperbaiki bug SPARQL tentang cakupan variabel dalam pola opsional.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.3.1.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.6.2`
- Gremlin versi terbaru didukung: `3.6.5`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.3.1.0

Anda dapat meningkatkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.3.0.0 (2023-11-15)

Pada 2023-11-15, versi mesin 1.3.0.0 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Note

Engine release 1.3.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.3.0.0 ke engine versi 1.3.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.3` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, `neptune1.2` atau. dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.3.0.0 ke atas. Demikian pula, Anda harus menggunakan grup parameter cluster 1.4.0.0 untuk versi mesin 1.4.0.0 dan di atasnya. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

Fitur Baru dalam Rilis Mesin Ini

- Merilis API data Neptunus.

  API data Amazon Neptunus menyediakan dukungan SDK untuk lebih dari 40 operasi data Neptunus, termasuk pemuatan data, eksekusi kueri, penyelidikan data, dan pembelajaran mesin. Ini mendukung ketiga bahasa query Neptunus (Gremlin, OpenCypher dan SPARQL), dan tersedia dalam semua bahasa SDK. Ini secara otomatis menandatangani permintaan API dan sangat menyederhanakan integrasi Neptunus ke dalam aplikasi Anda.
- Menambahkan dukungan untuk mengintegrasikan Tanpa OpenSearchServer dengan Neptunus.

Perbaikan dalam Rilis Mesin Ini

Perbaikan pembaruan mesin Neptunus

Neptunus telah mengubah cara merilis pembaruan mesin sehingga Anda dapat memiliki kontrol lebih besar atas proses pembaruan. Alih-alih merilis tambalan untuk perubahan yang tidak melanggar, Neptunus sekarang merilis versi minor yang dapat dikontrol menggunakan bidang `AutoMinorVersionUpgrade` instance, dan tentang mana Anda dapat menerima notifikasi dengan berlangganan acara tersebut. `RDS-EVENT-0156`

Lihat Memelihara Cluster DB Amazon Neptunus untuk informasi selengkapnya tentang perubahan ini.

Enkripsi dalam peningkatan transit

Neptunus tidak lagi mendukung cipher suite berikut:

- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

Neptunus hanya mendukung suite cipher kuat berikut dengan TLS 1.2:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`

Perbaikan Gremlin

- Menambahkan dukungan di mesin DFE untuk langkah-langkah Gremlin berikut:
  - `FoldStep`
  - `GroupStep`
  - `GroupCountStep`
  - `TraversalMapStep`
  - `UnfoldStep`
  - `LabelStep`
  - `PropertyKeyStep`

  - `PropertyValueStep`
  - `AndStep`
  - `OrStep`
  - `ConstantStep`
  - `CountGlobalStep`
- Kueri Gremlin DFE yang dioptimalkan berencana untuk menghindari pemindaian vertex penuh saat menggunakan modulasi. `by()`
- Peningkatan kinerja kardinalitas rendah dan kueri latensi rendah secara signifikan.
- Menambahkan dukungan DFE untuk predikat TinkerPop `Or` filter.
- Peningkatan dukungan DFE traversal untuk filter pada kunci yang sama, untuk kueri seperti berikut:

```
g.withSideEffect("Neptune#useDFE", true)
 .V()
 .has('name', 'marko')
 .and(
   or(
     has('name', eq("marko")),
     has('name', eq("vardas"))
   )
 )
```

- Peningkatan penanganan kesalahan untuk `fail()` langkah tersebut.

Perbaikan OpenCypher

- Peningkatan kinerja kardinalitas rendah dan kueri latensi rendah secara signifikan.
- Peningkatan kinerja perencanaan kueri ketika query berisi banyak jenis node.
- Mengurangi latensi semua kueri VLP.
- Peningkatan kinerja dengan menghapus sambungan pipa redundan untuk kueri pola simpul tunggal.
- Peningkatan kinerja untuk kueri yang berisi pola multi-hop dengan siklus, seperti ini:

```
MATCH (n)-->()-->()-->(m)
RETURN n m
```

Perbaikan SPARQL

- Memperkenalkan operator SPARQL baru:. `PipelineHashIndexJoin`
- Peningkatan kinerja validasi URI untuk kueri SPARQL.
- Peningkatan kinerja permintaan pencarian teks lengkap SPARQL dengan menyelesaikan istilah kamus secara batch.

Perbaikan Cacat dalam Rilis Mesin Ini

Perbaikan Gremlin

- Memperbaiki bug Gremlin di mana kebocoran transaksi akan terjadi saat memeriksa titik akhir status kueri Gremlin untuk kueri dengan predikat di lintasan anak untuk langkah-langkah yang tidak diproses secara asli di mesin DFE.
- Memperbaiki bug Gremlin yang tidak `valueMap()` dioptimalkan di mesin DFE di bawah traversal. `by()`
- Memperbaiki bug Gremlin di mana label langkah yang dilampirkan tidak `UnionStep` disebarkan ke elemen jalur terakhir dari lintasan turunannya masing-masing.
- Memperbaiki bug Gremlin di mana kueri akan gagal karena berisi terlalu banyak TinkerPop langkah dan kemudian tidak akan dibersihkan.
- Memperbaiki bug Gremlin di mana a `NullPointerException` akan dilemparkan `mergeV` dan `mergeE` melangkah.
- Memperbaiki bug Gremlin di mana tidak `order()` akan mengurutkan output string dengan benar ketika beberapa di antaranya berisi karakter spasi.
- Memperbaiki masalah kebenaran Gremlin yang terjadi saat `valueMap` langkah diproses di mesin DFE.
- Memperbaiki masalah kebenaran Gremlin yang terjadi saat `GroupStep` atau bersarang di `GroupCountStep` traversal kunci.

Perbaikan OpenCypher

- Memperbaiki bug OpenCypher yang melibatkan penanganan kesalahan di sekitar karakter NULL.
- Memperbaiki bug dalam penanganan transaksi OpenCypher Bolt.

Perbaikan SPARQL

- Memperbaiki bug SPARQL di mana nilai di dalam fungsi rekursif tidak diselesaikan dengan benar.
- Memperbaiki bug SPARQL yang menyebabkan penurunan kinerja ketika sejumlah besar nilai disuntikkan menggunakan klausa. `VALUES`
- Memperbaiki bug SPARQL di mana panggilan ke `REGEX` operator pada literal dengan tag bahasa tidak akan pernah berhasil.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.3.0.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.6.2`
- Gremlin versi terbaru didukung: `3.6.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.3.0.0

Anda dapat memutakhirkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.1.2 (2024-08-05)

Pada 2024-08-05, engine versi 1.2.1.2 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:

- Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
- Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

  Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

  Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.
- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Warning

Masalah terdeteksi dengan SPARQL 1.1 Graph Store HTTP Protocol (GSP) yang mungkin ada dalam kondisi tertentu ketika GSP digunakan dengan kebijakan otorisasi berbasis

tindakan. Jika Anda menggunakan Protokol HTTP Toko Grafik SPARQL 1.1 dengan kebijakan otorisasi berbasis tindakan, kami sarankan untuk meningkatkan ke versi mesin Neptunus minor terbaru (setidaknya 1.2.1.2) yang menyertakan perbaikan untuk masalah ini.

Perbaikan Cacat dalam Rilis Mesin Ini

Perbaikan SPARQL

- Memperbaiki masalah dengan SPARQL 1.1 Graph Store HTTP Protocol (GSP) yang mungkin ada dalam kondisi tertentu ketika GSP digunakan dengan kebijakan otorisasi berbasis tindakan.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.1.2, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.6.2`
- Gremlin versi terbaru didukung: `3.6.2`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.1.2

Anda dapat memutakhirkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
```

```
    --engine-version 1.2.1.2 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.2 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.1.1 (2024-03-11)

Pada 2024-03-11, engine versi 1.2.1.1 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:
>
> - Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
> - Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.
>
> Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.
>
> Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.
> - Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/`

`openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

Perbaikan umum

Neptunus telah meningkatkan peringatan yang ditunjukkan di profil/jelaskan.

Perbaikan Gremlin

- Perhitungan statistik DFE yang ditingkatkan untuk menghindari instans Tanpa Server yang sangat tinggi NCUs .
- Peningkatan kinerja Gremlin untuk WITHIN.

Perbaikan Cacat dalam Rilis Mesin Ini

Perbaikan Gremlin

- Perbaikan bug dengan pemesanan paket kueri mesin Gremlin DFE.
- Perbaikan bug dengan out-of-memory kesalahan Gremlin saat awalnya dilaporkan sebagai. InternalFailureException
- Perbaikan bug di mana IAM ARN tidak ada dalam log audit untuk permintaan koneksi websocket awal yang berhasil.
- Perbaikan bug untuk kueri Gremlin dengan TinkerPop sesi diaktifkan ketika kueri dalam sesi gagal bahkan ketika semuanya hanya dibaca dan terhubung ke instance pembaca.

Perbaikan OpenCypher

- Perbaikan bug dalam klausa SET OpenCypher untuk memungkinkan pengaturan pada ekspresi non-variabel (yaitu: match (n: test) set (kasus ketika n.prop = 2 lalu n end) .prop = 3 return n.prop.
- Perbaikan bug untuk gagal kueri OpenCypher yang melibatkan agregasi dan urutan oleh.
- Peningkatan UNWIND dari daftar besar yang berisi peta statis.
- Perbaikan bug OpenCypher MERGE query menggunakan id kustom dengan nilai duplikat.

Perbaikan SPARQL

- Perbaikan bug di perencana kueri SPARQL DFE.
- Perbaikan bug untuk SPARQL saat digunakan dengan kata kunci BIND dan OPSIONAL.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan kluster DB ke versi 1.2.1.1, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.6.2`
- Gremlin versi terbaru didukung: `3.6.2`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.1.1

Anda dapat memutakhirkan ke rilis ini dari rilis mesin 1.2.0.0 atau lebih tinggi.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.1 ^
```

```
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.1.0 (2023-03-08)

Pada 2023-03-08, engine versi 1.2.1.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:
>
> - Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan

keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

- Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

  Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

  Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.
- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis: 1.2.1.0.R2 (2023-05-02)
- Rilis: 1.2.1.0.R3 (2023-06-13)
- Rilis: 1.2.1.0.R4 (2023-08-10)
- Rilis: 1.2.1.0.R5 (2023-09-02)
- Rilis: 1.2.1.0.R6 (2023-09-12)
- Rilis: 1.2.1.0.R7 (2023-10-06)

Fitur Baru dalam Rilis Mesin Ini

- Menambahkan dukungan untuk TinkerPop 3.6.2, yang menambahkan banyak fitur Gremlin baru seperti yang barumergeV(),,mergeE(), element() dan langkah-langkah. fail() mergeE()Langkah-langkah mergeV() dan adalah catatan khusus karena mereka menawarkan opsi deklaratif yang telah lama ditunggu-tunggu untuk melakukan operasi seperti upsert, yang seharusnya sangat menyederhanakan pola kode yang ada dan membuat Gremlin lebih mudah dibaca. Versi 3.6.x juga menambahkan predikat regex, kelebihan beban baru ke property() langkah yang mengambilMap, dan revisi besar perilaku by() modulasi yang jauh lebih konsisten di semua langkah yang menggunakannya.

  Lihat log TinkerPop perubahan dan halaman pemutakhiran untuk informasi tentang perubahan di versi 3.6 dan hal-hal yang perlu dipertimbangkan saat memutakhirkan.

  Jika Anda menggunakan fold().coalesce(unfold(), <mutate>) untuk sisipan bersyarat, kami sarankan Anda bermigrasi ke mergeV/E() sintaks baru, dijelaskan di sini dan di sini. Neptunus menggunakan pola penguncian yang lebih sempit Merge untuk daripada Coalesce for, yang dapat mengurangi pengecualian modifikasi bersamaan (). CMEs

  Untuk informasi lebih lanjut tentang fitur baru yang tersedia dalam TinkerPop rilis ini, lihat blog Stephen Mallette, Menjelajahi fitur baru Apache TinkerPop 3.6.x di Amazon Neptunus.
- Menambahkan dukungan untuk tipe instans R6i, didukung oleh prosesor Intel Xeon Scalable generasi ke-3. Ini sangat cocok untuk beban kerja intensif memori dan menawarkan compute/price kinerja hingga 15% lebih baik dan bandwidth memori per vCPU hingga 20% lebih tinggi daripada jenis instans R5 yang sebanding.
- Menambahkan titik akhir API ringkasan grafik untuk grafik properti dan grafik RDF, yang memungkinkan Anda mendapatkan laporan ringkasan cepat tentang grafik Anda.

  Untuk grafik properti (PG), API ringkasan grafik menyediakan daftar read-only label node dan edge dan kunci properti, bersama dengan jumlah node, tepi, dan properti. Untuk grafik RDF, ini menyediakan daftar kelas dan kunci predikat, bersama dengan jumlah paha depan, subjek, dan predikat.

  Perubahan berikut sejalan dengan API ringkasan grafik baru:

  - Menambahkan aksi GetGraphSummarydataplane baru.
  - Menambahkan rdf/statistics endpoint baru untuk menggantikan sparql/statistics endpoint, yang sekarang sudah usang.

  - Mengubah nama `summary` bidang dalam respons status statistik`signatureInfo`, agar tidak membingungkan dengan informasi ringkasan grafik. Versi mesin sebelumnya terus digunakan `summary` dalam respons JSON.
  - Mengubah ketepatan `date` bidang dalam respons status statistik dari menit ke milidetik. Format sebelumnya adalah `2020-05-07T23:13Z` (presisi menit), sedangkan format baru adalah `2023-01-24T00:47:43.319Z` (presisi milidetik). Keduanya sesuai dengan ISO 8601, tetapi perubahan ini dapat merusak kode yang ada, tergantung pada bagaimana tanggal diuraikan.
  - Menambahkan keajaiban `%statistics`baris baru di Workbench yang memungkinkan Anda mengambil statistik mesin DFE.
  - Menambahkan sihir `%summary`baris baru di Workbench yang memungkinkan Anda mengambil informasi ringkasan grafik.
- Menambahkan pencatatan kueri lambat ke kueri log yang membutuhkan waktu lebih lama untuk dieksekusi daripada ambang batas yang ditentukan. Anda mengaktifkan dan mengontrol logging query lambat menggunakan dua parameter dinamis baru, yaitu neptune_enable_slow_query_log, dan neptune_slow_query_log_threshold.
- Menambahkan dukungan untuk dua parameter dinamis, yaitu parameter cluster baru, neptune_enable_slow_query_log, dan neptune_slow_query_log_threshold. Ketika Anda membuat perubahan ke parameter dinamis, itu akan segera berlaku, tanpa memerlukan reboot instance apa pun.
- Menambahkan fungsi OpenCypher removeKeyFromMap () khusus Neptunus yang menghapus kunci tertentu dari peta dan mengembalikan peta baru yang dihasilkan.

Perbaikan dalam Rilis Mesin Ini

- Memperluas dukungan Gremlin DFE untuk `limit` langkah-langkah dengan cakupan lokal.
- Menambahkan dukungan `by()` modulasi untuk Gremlin `DedupGlobalStep` di mesin DFE.
- Menambahkan dukungan DFE untuk Gremlin `SelectStep` dan. `SelectOneStep`
- Peningkatan kinerja dan perbaikan kebenaran untuk berbagai operator Gremlin, termasuk`repeat`,,, dan. `coalesce` `store` `aggregate`
- Peningkatan kinerja kueri OpenCypher yang melibatkan dan. `MERGE` `OPTIONAL MATCH`
- Peningkatan kinerja query OpenCypher yang melibatkan `UNWIND` daftar peta nilai literal.
- Peningkatan kinerja kueri OpenCypher yang memiliki filter untuk. `IN` `id` Misalnya:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Menambahkan kemampuan untuk menentukan IRI dasar untuk kueri SPARQL menggunakan pernyataan BASE (lihat). Basis IRI default untuk kueri dan pembaruan
- Mempersingkat waktu tunggu pemrosesan beban untuk beban curah khusus tepi Gremlin dan OpenCypher.
- Pemuatan massal yang dibuat dilanjutkan secara asinkron saat Neptunus memulai ulang untuk menghindari waktu tunggu yang lama yang disebabkan oleh masalah konektivitas Amazon S3 sebelum gagal melanjutkan upaya.
- Peningkatan penanganan kueri SPARQL DESCRIBE yang memiliki petunjuk kueri DescribeMode yang disetel ke `"CBD"` (deskripsi terbatas ringkas) dan yang melibatkan sejumlah besar node kosong.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug OpenCypher di mana kueri mengembalikan string,`"null"`, bukan nilai null di Bolt dan SPARQL-JSON.
- Memperbaiki bug OpenCypher dalam pemahaman daftar yang menghasilkan nilai nol daripada nilai yang disediakan untuk elemen daftar.
- Memperbaiki bug OpenCypher di mana nilai byte tidak diserialkan dengan benar.
- Memperbaiki bug Gremlin `UnionStep` yang terjadi ketika input adalah tepi yang melintasi ke simpul di dalam traversal anak.
- Memperbaiki bug Gremlin yang menyebabkan label langkah yang terkait dengan `UnionStep` tidak menyebar dengan benar ke langkah terakhir dari setiap traversal anak.
- Memperbaiki bug Gremlin untuk `dedup` langkah dengan label mengikuti `repeat` langkah, di mana label yang dilampirkan pada `dedup` langkah tidak tersedia untuk digunakan dalam kueri lebih lanjut.
- Memperbaiki bug Gremlin di mana menerjemahkan `repeat` langkah di dalam `union` langkah gagal dengan kesalahan internal.
- Memperbaiki masalah kebenaran Gremlin untuk kueri DFE dengan `limit` sebagai traversal anak dari langkah-langkah non-serikat dengan kembali ke Tinkerpop. Kueri dalam bentuk seperti ini terpengaruh:

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- Memperbaiki bug SPARQL di mana `SPARQL GRAPH` pola tidak akan mempertimbangkan kumpulan data yang disediakan oleh klausa. `FROM NAMED`
- Memperbaiki bug SPARQL di mana SPARQL `DESCRIBE` dengan beberapa `FROM` and/or `FROM NAMED` klausa tidak selalu menggunakan data dengan benar dari grafik default dan terkadang melemparkan pengecualian. Lihat Perilaku DESKRIPSI SPARQL sehubungan dengan grafik default.
- Memperbaiki bug SPARQL sehingga pesan pengecualian yang benar dikembalikan ketika karakter null ditolak.
- Memperbaiki bug penjelasan SPARQL yang memengaruhi paket yang berisi operator. PipelinedHashIndexJoin
- Memperbaiki bug yang menyebabkan kesalahan internal dilemparkan saat kueri yang mengembalikan nilai konstan dikirimkan.
- Memperbaiki masalah dengan logika detektor kebuntuan yang terkadang membuat mesin tidak responsif.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.1.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.6.2`
- Gremlin versi terbaru didukung: `3.6.2`
- Versi OpenCypher: `Neptune-9.0.20190305-1.1`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.1.0

Anda dapat memutakhirkan ke rilis ini secara manual dari rilis mesin Neptunus sebelumnya yang lebih besar dari atau sama dengan 1.1.0.0.

> **Note**
>
> Dimulai dengan rilis mesin 1.2.0.0, semua grup parameter khusus dan grup parameter cluster khusus yang Anda gunakan dengan versi mesin lebih awal dari sekarang `1.2.0.0` harus dibuat ulang menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis dari `1.2.0.0` seterusnya. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

Anda tidak akan secara otomatis ditingkatkan ke rilis versi utama ini.

Peningkatan ke Rilis Ini

Amazon Neptune 1.2.1.0 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.1.0.R7 (2023-10-06)

Pada 2023-10-06, versi mesin 1.2.1.0.R7 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Note

Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:

- Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
- Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

  Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke

jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.

- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, pengaturan jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug di mana dalam beberapa kasus transaksi yang gagal tidak ditutup dengan benar.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.1.0.R7, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.6.2`
- Gremlin versi terbaru didukung: `3.6.2`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Peningkatan ke Rilis Ini

Amazon Neptune 1.2.1.0.R7 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.1.0.R6 (2023-09-12)

Pada 2023-09-12, versi mesin 1.2.1.0.R6 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:
>
> - Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan

keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

- Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

  Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

  Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.
- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Fitur Baru dalam Rilis Mesin Ini

- Merilis API data Neptunus.

  API data Amazon Neptune menyediakan dukungan SDK untuk memuat data, menjalankan kueri, mendapatkan informasi tentang data Anda, dan menjalankan operasi pembelajaran mesin. Ini mendukung bahasa query Gremlin dan OpenCypher di Neptunus dan tersedia dalam semua bahasa SDK. Ini secara otomatis menandatangani permintaan API dan sangat menyederhanakan integrasi Neptunus ke dalam aplikasi.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug yang dapat menyebabkan lonjakan CPU di bawah beban tinggi saat log Kueri Lambat diaktifkan.
- Memperbaiki bug Gremlin di mana menambahkan tepi dan propertinya diikuti oleh `inV()` atau `outV()` memunculkan file. `InternalFailureException`
- Memperbaiki beberapa masalah dengan rantai peran IAM yang menyebabkan penurunan kinerja bulk-loader dalam beberapa kasus.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.1.0.R6, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.6.2`
- Gremlin versi terbaru didukung: `3.6.2`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Peningkatan ke Rilis Ini

Amazon Neptunus 1.2.1.0.R6 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.1.0.R5 (2023-09-02)

Pada 2023-09-02, versi mesin 1.2.1.0.R5 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:
>
> - Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
> - Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum

peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.

- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, pengaturan jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Fitur Baru dalam Rilis Mesin Ini

- Merilis API data Neptunus.

  API data Amazon Neptune menyediakan dukungan SDK untuk memuat data, menjalankan kueri, mendapatkan informasi tentang data Anda, dan menjalankan operasi pembelajaran mesin. Ini mendukung bahasa query Gremlin dan OpenCypher di Neptunus dan tersedia dalam semua bahasa SDK. Ini secara otomatis menandatangani permintaan API dan sangat menyederhanakan integrasi Neptunus ke dalam aplikasi.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin di mana menambahkan tepi dan propertinya diikuti oleh `inV()` atau `outV()` memunculkan file. `InternalFailureException`

- Memperbaiki beberapa masalah dengan rantai peran IAM yang menyebabkan penurunan kinerja bulk-loader dalam beberapa kasus.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.1.0.R5, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.6.2`
- Gremlin versi terbaru didukung: `3.6.2`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Peningkatan ke Rilis Ini

Amazon Neptune 1.2.1.0.R5 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui Dukungan AWS Premium.

Mesin Amazon Neptunus Versi 1.2.1.0.R4 (2023-08-10)

Pada 2023-08-10, versi mesin 1.2.1.0.R4 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Important

Perubahan yang terjadi pada pelepasan mesin ini dalam beberapa kasus dapat menyebabkan Anda mengamati kinerja beban curah yang menurun. Akibatnya, upgrade ke rilis ini telah ditangguhkan sementara sampai masalah telah diselesaikan.

Note

Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:

- Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
- Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan,

upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.

- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

- Tambahkan dukungan Graphson-1.0 untuk Gremlin. Untuk menggunakan Graphson-1.0, lulus `Accept header` dengan nilai:

```
application/vnd.gremlin-v1.0+json;types=false
```

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin di mana kebocoran transaksi akan terjadi saat memeriksa titik akhir status kueri Gremlin untuk kueri dengan predikat di lintasan anak untuk langkah-langkah yang tidak diproses secara asli.
- Memperbaiki bug OpenCypher dalam penanganan transaksi Bolt.
- Memperbaiki masalah konkurensi pada lapisan penyimpanan yang dapat menyebabkan crash.
- Memperbaiki bug di log kueri lambat untuk memastikan bahwa mereka tidak aktif saat dinonaktifkan.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.1.0.R4, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.6.2`
- Gremlin versi terbaru didukung: `3.6.5`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.1.0.R4

Peningkatan ke Rilis Ini

Amazon Neptunus 1.2.1.0.R4 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.1.0.R3 (2023-06-13)

Pada 2023-06-13, versi mesin 1.2.1.0.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Important

Perubahan yang terjadi pada pelepasan mesin ini dalam beberapa kasus dapat menyebabkan Anda mengamati kinerja beban curah yang menurun. Akibatnya, upgrade ke rilis ini telah ditangguhkan sementara sampai masalah telah diselesaikan.

Note

Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:

- Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
- Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan,

upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.

- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Fitur Baru dalam Rilis Mesin Ini

- Menambahkan dukungan untuk pemuatan massal lintas akun menggunakan rantai peran IAM.

Perbaikan dalam Rilis Mesin Ini

- Meningkatkan `fail()` langkah Gremlin untuk membedakan pengecualian yang dihasilkannya dari generik `InternalFailureException` dan untuk memastikan bahwa setiap pesan yang diberikan pengguna yang diberikan padanya disebarkan kembali ke penelepon.
- Peningkatan optimasi mesin kueri Gremlin untuk`store`,,,`aggregate`, `cap` dan. `limit` `hasLabel`
- Ditambahkan dukungan untuk fungsi trignometrik OpenCypher:
  - `acos()`
  - `asin()`
  - `atan()`
  - `atan2()`
  - `cos()`

  - `cot()`
  - `degrees()`
  - `pi()`
  - `radians()`
  - `sin()`
  - `tan()`
- Ditambahkan dukungan untuk beberapa fungsi agregasi OpenCypher:
  - `percentileDisc()`
  - `stDev()`
- Ditambahkan dukungan untuk `epochmillis()` fungsi OpenCypher yang mengkonversi ke. `datetime` `epochmillis` Misalnya:

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

- Ditambahkan dukungan untuk OpenCypher modulo () operator. `%`
- Menambahkan dukungan untuk alat OpenCypher Static Debug Explain.
- Ditambahkan dukungan untuk fungsi OpenCypher`randomUUID()`.
- Peningkatan kinerja OpenCypher:
  - Meningkatkan parser dan query planner.
  - Peningkatan pemanfaatan CPU di mesin DFE.
  - Meningkatkan kinerja kueri yang berisi beberapa klausa pembaruan yang menggunakan kembali variabel yang sama. Contohnya adalah:

```
MERGE (n {name: 'John'})
  or
MERGE (m {name: 'Jim'})
  or
MERGE (n)-[:knows {since: 2023}]#(m)
```

  - Paket kueri yang dioptimalkan untuk pola kueri multi-hop seperti:

```
MATCH (n)-->()-->()-->(m)
RETURN n m
```

  - Meningkatkan kinerja daftar dan injeksi peta melalui kueri parameter. Misalnya:

```
UNWIND $idList as id MATCH (n {`~id`: id})
RETURN n.name
```

  - Peningkatan eksekusi query yang mengandung `WITH` dengan menjadikannya penghalang yang tepat.
  - Dioptimalkan untuk menghindari perwujudan nilai dalam `Unfold` dan fungsi agregasi yang berlebihan.
- Meningkatkan kinerja query SPARQL yang berisi sejumlah besar input statis dalam `VALUES` klausa, seperti:

```
SELECT ?n WHERE { VALUES (?name) { ("John") ("Jim") ... many values ... } ?n a ?
n_type . ?n ?name . }
```

- Peningkatan kinerja kueri CBD SPARQL.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin di mana kueri panjang dengan deep nesting menyebabkan penggunaan CPU yang tinggi dan batas waktu kueri selama fase perencanaan kueri.
- Memperbaiki bug Gremlin di mana yang tidak valid `NullPointerException` dapat dilemparkan saat menggunakan atau. `mergeV` `mergeE`

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.1.0.R3, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.6.2`
- Gremlin versi terbaru didukung: `3.6.2`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.1.0.R3

Peningkatan ke Rilis Ini

Amazon Neptunus 1.2.1.0.R3 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.1.0.R2 (2023-05-02)

Pada 2023-05-02, versi mesin 1.2.1.0.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:
>
> - Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
> - Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.
>
> Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.
>
> Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.
> - Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/`

`openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

- Menambahkan `enableInterContainerTrafficEncryption` parameter ke semua Neptunus APIs Neptunus, yang dapat Anda gunakan untuk mengaktifkan dan menonaktifkan enkripsi lalu lintas antar-kontainer dalam pelatihan atau pekerjaan penyetelan hiper-parameter.
- Menambahkan dukungan multi-label untuk Gremlin `mergeV()` dan. `mergeE()`

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug OpenCypher di mana kueri pembaruan dan pengembalian tidak ditangani`orderBy`,, `limit` atau dengan benar. `skip`
- Memperbaiki bug OpenCypher yang memungkinkan parameter yang terkandung dalam satu permintaan diganti oleh parameter yang terkandung dalam permintaan simultan lainnya.
- Memperbaiki bug OpenCypher di mana log kueri lambat tidak berisi waktu kueri yang benar.
- Memperbaiki bug Gremlin di mana kebocoran transaksi dapat terjadi ketika kueri yang berisi `GroupCountStep` dikirimkan sebagai string.
- Memperbaiki bug Gremlin di mana WebSocket kueri gagal saat log kueri lambat diaktifkan.
- Memperbaiki bug Gremlin di mana log debug penghitung penyimpanan hilang di log kueri lambat untuk permintaan. WebSocket
- Memperbaiki beberapa bug Gremlin yang melibatkan `mergeV()` dan. `mergeE()`
- Memperbaiki bug SPARQL di mana biaya kueri grafik bernama salah estimasi, yang mengarah ke rencana dan kesalahan kueri yang kurang optimal. out-of-memory
- Memperbaiki bug yang memengaruhi otorisasi untuk kueri Gremlin dan OpenCypher pada cluster yang mendukung IAM.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.1.0.R2, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.6.2`

- Gremlin versi terbaru didukung: `3.6.2`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.1.0.R2

Peningkatan ke Rilis Ini

Amazon Neptunus 1.2.1.0.R2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.0.2 (2022-11-20)

Pada 2022-11-20, engine versi 1.2.0.2 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:
>
> - Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
> - Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.
>
>   Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.
>
>   Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.

- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis: 1.2.0.2.R2 (2022-12-15)
- Rilis: 1.2.0.2.R3 (2023-03-27)
- Rilis: 1.2.0.2.R4 (2023-05-08)
- Rilis: 1.2.0.2.R5 (2023-08-16)
- Rilis: 1.2.0.2.R6 (2023-09-12)

Fitur Baru dalam Rilis Mesin Ini

- Memperkenalkan inferensi induktif real-time untuk Gremlin di Neptunus ML.
- Memperkenalkan ekstensi OpenCypher yang mendukung penentuan nilai ID khusus untuk entitas alih-alih yang dihasilkan UUIDs Neptunus. Kemampuan untuk menetapkan kustom IDs membuatnya lebih mudah untuk bermigrasi ke Neptunus dari Neo4j.

  Warning

  Ekstensi ke spesifikasi OpenCypher ini tidak kompatibel ke belakang, karena sekarang `~id` dianggap sebagai nama properti yang dicadangkan. Jika Anda sudah menggunakan `~id` sebagai properti dalam data dan kueri, Anda harus memigrasikan `~id` properti ke kunci properti baru sebelum memutakhirkan ke rilis ini.

- Menambahkan beberapa `DESCRIBE` mode SPARQL baru bersama dengan petunjuk kueri untuk mengonfigurasinya.

Perbaikan dalam Rilis Mesin Ini

- Peningkatan kinerja OpenCypher, terutama untuk kueri VLP.
- Peningkatan kinerja DFE untuk kueri Gremlin dengan batas non-terminal, seperti:

```
g.withSideEffect('Neptune#useDFE',true).V().hasLabel('Student').limit(5).out('takesCourse')
```

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.0.2, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.0.2

Peningkatan ke Rilis Ini

Amazon Neptune 1.2.0.2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.0.2.R6 (2023-09-12)

Pada 2023-09-12, versi mesin 1.2.0.2.R6 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:
>
> - Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
> - Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum

peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.

- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug SPARQL di mana `REGEX` operator tidak akan pernah berhasil saat dipanggil pada literal yang diberi tag bahasa.
- Memperbaiki masalah yang menyebabkan kinerja beban massal menurun.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.0.2.R6, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.5`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.0.2.R6

Cluster DB Neptunus Anda akan ditingkatkan ke rilis patch pemeliharaan ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi engine. `1.2.0.2`

Peningkatan ke Rilis Ini

Amazon Neptune 1.2.0.2.R6 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.0.2.R5 (2023-08-16)

Pada 2023-08-16, versi mesin 1.2.0.2.R5 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Important

Perubahan yang terjadi pada pelepasan mesin ini dalam beberapa kasus dapat menyebabkan Anda mengamati kinerja beban curah yang menurun. Akibatnya, upgrade ke rilis ini telah ditangguhkan sementara sampai masalah telah diselesaikan.

Note

Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:

- Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
- Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

  Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.

- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin di mana tidak `order()` akan mengurutkan output string dengan benar ketika beberapa di antaranya berisi karakter spasi.
- Memperbaiki bug Gremlin di mana kebocoran transaksi akan terjadi saat memeriksa titik akhir status kueri Gremlin untuk kueri dengan predikat di lintasan anak untuk langkah-langkah yang tidak diproses secara asli.
- Memperbaiki bug OpenCypher dalam penanganan transaksi Bolt.
- Memperbaiki masalah konkurensi pada lapisan penyimpanan yang dapat menyebabkan crash.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.0.2.R5, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.5`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.0.2.R5

Cluster DB Neptunus Anda akan ditingkatkan ke rilis patch pemeliharaan ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi engine. `1.2.0.2`

Peningkatan ke Rilis Ini

Amazon Neptune 1.2.0.2.R5 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.0.2.R4 (2023-05-08)

Pada 2023-05-08, versi mesin 1.2.0.2.R4 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Note

Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:

- Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
- Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

  Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

  Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.
- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug SPARQL di mana sejumlah besar nilai yang disuntikkan melalui `VALUES` klausa dapat menyebabkan penurunan kinerja.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.0.2.R4, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.6`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.0.2.R4

Cluster DB Neptunus Anda akan ditingkatkan ke rilis patch pemeliharaan ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi engine. `1.2.0.2`

Peningkatan ke Rilis Ini

Amazon Neptune 1.2.0.2.R4 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.0.2.R3 (2023-03-27)

Pada 2023-03-27, versi mesin 1.2.0.2.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:
>
> - Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
> - Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum

peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.

- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

- Untuk cluster DB tanpa server, ubah pengaturan kapasitas minimum menjadi 1.0 NCU, dan pengaturan maksimum valid terendah menjadi 2,5. NCUs Lihat Penskalaan kapasitas dalam cluster DB Neptunus Tanpa Server
- Menambahkan `enableInterContainerTrafficEncryption` parameter ke semua Neptunus APIs Neptunus, yang dapat Anda gunakan untuk mengaktifkan dan menonaktifkan enkripsi lalu lintas antar-kontainer dalam pelatihan atau pekerjaan penyetelan hiper-parameter.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin yang tidak dikenali sebagai `option(Predicate)` sintaks Gremlin yang valid.
- Memperbaiki bug Gremlin yang menyebabkan kueri tidak dibersihkan dengan benar jika gagal karena mengandung terlalu banyak langkah.

- Memperbaiki masalah kebenaran Gremlin yang memengaruhi kueri DFE `limit` sebagai traversal anak dari langkah non-serikat dengan kembali ke Tinkerpop. Berikut adalah contoh kueri seperti itu:

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- Memperbaiki potensi kebocoran transaksi Gremlin saat kueri dikirimkan sebagai String berisi. `GroupCountStep`
- Memperbaiki bug OpenCypher di mana jenis nilai parameter tidak disimpulkan dengan benar dalam daftar atau daftar peta.
- Memperbaiki bug OpenCypher di mana kueri pembaruan dan pengembalian tidak ditangani`orderBy`,, `limit` atau dengan benar. `skip`
- Memperbaiki bug OpenCypher yang memungkinkan parameter yang terkandung dalam satu permintaan diganti oleh parameter yang terkandung dalam permintaan simultan lainnya.
- Memperbaiki bug SPARQL di mana sejumlah besar nilai yang disuntikkan dalam `VALUES` klausa dapat menyebabkan penurunan kinerja.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.0.2.R3, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.6`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.0.2.R3

Cluster DB Neptunus Anda akan ditingkatkan ke rilis patch pemeliharaan ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi engine. `1.2.0.2`

Peningkatan ke Rilis Ini

Amazon Neptune 1.2.0.2.R3 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.0.2.R2 (2022-12-15)

Pada 2022-12-15, versi mesin 1.2.0.2.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Note

Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:

- Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
- Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

  Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

  Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.
- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, pengaturan jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

- Peningkatan kinerja kueri OpenCypher yang melibatkan dan. `MERGE` `OPTIONAL MATCH`
- Peningkatan kinerja query OpenCypher yang melibatkan `UNWIND` daftar peta nilai literal.
- Peningkatan kinerja kueri OpenCypher yang memiliki filter untuk. `IN` `id` Misalnya:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Peningkatan kinerja dan perbaikan kebenaran untuk berbagai operator Gremlin, termasuk`repeat`,,, dan. `coalesce` `store` `aggregate`

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug OpenCypher di mana kueri mengembalikan string,`"null"`, bukan nilai null di Bolt dan SPARQL-JSON.
- Memperbaiki bug Gremlin yang menyebabkan label langkah yang dilampirkan `UnionStep` tidak disebarkan ke elemen jalur terakhir dari lintasan turunannya.
- Memperbaiki bug Gremlin yang menyebabkan `valueMap()` tidak dioptimalkan di bawah `by()` traversal di mesin DFE.
- Memperbaiki bug Gremlin di mana kueri baca yang dieksekusi sebagai bagian dari transaksi Gremlin yang lebih lama tidak akan mengunci baris.
- Memperbaiki bug log audit yang menyebabkan informasi yang tidak perlu dicatat dan bidang tertentu hilang dari log.
- Memperbaiki bug log audit di mana IAM ARN permintaan HTTP ke kluster DB berkemampuan IAM tidak direkam.
- Memperbaiki bug lookup-cache sehingga dapat membatasi memori tambahan yang digunakan untuk menulis ke cache.
- Memperbaiki bug lookup-cache yang melibatkan pengaturan mode hanya-baca untuk cache pencarian saat penulisan gagal.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.0.2.R2, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.0.2.R2

Cluster DB Neptunus Anda akan ditingkatkan ke rilis patch pemeliharaan ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi engine. `1.2.0.2`

Peningkatan ke Rilis Ini

Amazon Neptune 1.2.0.2.R2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.0.1 (2022-10-26)

Pada 2022-10-26, engine versi 1.2.0.1 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Note

Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:

- Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
- Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

  Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.

- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis Pemeliharaan: 1.2.0.1.R2 (2022-12-13)
- Rilis Pemeliharaan: 1.2.0.1.R3 (2023-09-27)

Fitur Baru dalam Rilis Mesin Ini

- Memperkenalkan Amazon Neptune Serverless, konfigurasi penskalaan otomatis sesuai permintaan yang meningkatkan skala cluster DB Anda untuk memenuhi peningkatan permintaan pemrosesan dan kemudian turun lagi saat permintaan menurun.

Perbaikan dalam Rilis Mesin Ini

- Peningkatan kinerja kueri Gremlin`order-by`. Kueri Gremlin dengan `order-by` di akhir `NeptuneGraphQueryStep` sekarang menggunakan ukuran potongan yang lebih besar untuk kinerja yang lebih baik. Ini tidak berlaku untuk `order-by` node internal (non-root) dari rencana kueri.
- Peningkatan kinerja kueri pembaruan Gremlin. Simpul dan tepi sekarang harus dikunci terhadap penghapusan sambil menambahkan tepi atau properti. Perubahan ini menghilangkan kunci duplikat dalam transaksi, yang meningkatkan kinerja.
- Peningkatan kinerja query Gremlin yang menggunakan `dedup()` bagian dalam `repeat()` subquery dengan menekan `dedup` ke bawah ke layer eksekusi asli.

- Menambahkan pesan kesalahan yang mudah digunakan untuk kesalahan otentikasi IAM. Pesan-pesan ini sekarang menunjukkan pengguna IAM Anda atau peran ARN, ARN sumber daya, dan daftar tindakan yang tidak sah untuk permintaan tersebut. Daftar tindakan yang tidak sah membantu Anda melihat apa yang mungkin hilang atau ditolak secara eksplisit dalam kebijakan IAM yang Anda gunakan.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin di mana menggunakan `PartitionStrategy` setelah memutakhirkan ke TinkerPop 3.5 secara tidak benar menghasilkan kesalahan dengan pesan, "PartitionStrategy tidak berfungsi dengan Traversals anonim," yang mencegah traversal dieksekusi.
- Memperbaiki bug kebenaran Gremlin yang melibatkan `WherePredicateStep` terjemahan, di mana mesin kueri Neptunus menghasilkan hasil yang salah untuk kueri yang digunakan dan variasinya. `where(P.neq('x'))`
- Memperbaiki bug OpenCypher di `MERGE` klausa yang dalam beberapa kasus menyebabkan duplikat node dan pembuatan edge.
- Memperbaiki bug SPARQL dalam penanganan kueri yang berisi `(NOT) EXISTS` dalam `OPTIONAL` klausa, di mana dalam beberapa kasus hasil kueri hilang.
- Memperbaiki bug pemuat massal yang menyebabkan regresi kinerja di bawah beban penyisipan berat.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.0.1, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.0.1

Peningkatan ke Rilis Ini

Amazon Neptune 1.2.0.1 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.0.1.R3 (2023-09-27)

Pada 2023-09-27, versi mesin 1.2.0.1.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:
>
> - Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
> - Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.
>
>   Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.
>
>   Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.
> - Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/`

`openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

- Menambahkan `enableInterContainerTrafficEncryption` parameter ke semua Neptunus APIs Neptunus, yang dapat Anda gunakan untuk mengaktifkan dan menonaktifkan enkripsi lalu lintas antar-kontainer dalam pelatihan atau pekerjaan penyetelan hiper-parameter.
- Untuk cluster DB tanpa server, ubah pengaturan kapasitas minimum menjadi 1.0 NCU, dan pengaturan maksimum valid terendah menjadi 2,5. NCUs Lihat Penskalaan kapasitas dalam cluster DB Neptunus Tanpa Server *(((before release, this change needs to be reflected in the serverless page too)))*.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug OpenCypher di mana update-and-return kueri tidak menangani`orderBy`,, `limit` atau dengan benar. `skip`
- Memperbaiki bug OpenCypher yang memungkinkan parameter yang terkandung dalam satu permintaan diganti oleh parameter yang terkandung dalam permintaan simultan lainnya.
- Memperbaiki bug OpenCypher dalam penanganan transaksi Bolt.
- Memperbaiki masalah kebenaran Gremlin untuk kueri DFE dengan `limit` sebagai traversal anak dari langkah-langkah non-serikat dengan kembali ke Tinkerpop. Misalnya, untuk kueri seperti ini:

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
 .select("a")
 .by(out()
 .limit(1))
```

- Memperbaiki bug Gremlin di mana kueri akan gagal karena berisi terlalu banyak TinkerPop langkah dan kemudian tidak akan dibersihkan.
- Memperbaiki bug Gremlin di mana tidak `order()` akan mengurutkan output string dengan benar ketika beberapa di antaranya berisi karakter spasi.
- Memperbaiki bug Gremlin di mana kebocoran transaksi dapat terjadi ketika kueri dikirimkan sebagai String dan berisi. `GroupCountStep`

- Memperbaiki bug Gremlin di mana kebocoran transaksi akan terjadi saat memeriksa titik akhir status kueri Gremlin untuk kueri dengan predikat di lintasan anak untuk langkah-langkah yang tidak diproses secara asli.
- Memperbaiki bug Gremlin tempat menambahkan Edge dan propertinya diikuti oleh `inV()` atau `outV()` menyebabkan file. `InternalFailureException`
- Memperbaiki masalah konkurensi di lapisan penyimpanan.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.0.1.R3, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.6`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.0.1.R3

Cluster DB Neptunus Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan engine versi 1.2.0.1

Peningkatan ke Rilis Ini

Amazon Neptune 1.2.0.1.R3 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.0.1.R2 (2022-12-13)

Pada 2022-12-13, versi mesin 1.2.0.1.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:
>
> - Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
> - Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum

peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.

- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, pengaturan jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

- Meningkatkan kinerja query OpenCypher yang melibatkan `UNWIND` pada daftar peta nilai literal.
- Peningkatan kinerja dan perbaikan kebenaran untuk berbagai operator Gremlin, termasuk`repeat`,,, dan. `coalesce` `store` `aggregate`

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug OpenCypher di mana kueri mengembalikan string,`"null"`, bukan nilai null di Bolt dan SPARQL-JSON.
- Memperbaiki bug log audit yang menyebabkan informasi yang tidak perlu dicatat dan bidang tertentu hilang dari log.
- Memperbaiki bug lookup-cache sehingga dapat membatasi memori tambahan yang digunakan untuk menulis ke cache.

- Memperbaiki bug lookup-cache yang melibatkan pengaturan mode hanya-baca untuk cache pencarian saat penulisan gagal.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.0.1.R2, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.0.1.R2

Cluster DB Neptunus Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan engine versi 1.2.0.1

Peningkatan ke Rilis Ini

Amazon Neptune 1.2.0.1.R2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
```

```
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.0.0 (2022-07-21)

Pada 2022-07-21, engine versi 1.2.0.0 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Note

Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:

- Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
- Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan,

upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.

- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, pengaturan jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis: 1.2.0.0.R2 (2022-10-14)
- Rilis: 1.2.0.0.R3 (2022-12-15)
- Rilis: 1.2.0.0.R4 (2023-09-29)

Fitur Baru dalam Rilis Mesin Ini

- Menambahkan dukungan untuk database global. Database global Neptunus mencakup Wilayah AWS beberapa, dan terdiri dari cluster DB primer di satu wilayah, dan hingga lima cluster DB sekunder di wilayah lain.
- Menambahkan dukungan untuk kontrol akses yang lebih terperinci dalam kebijakan IAM Neptunus daripada yang telah tersedia sebelumnya, berdasarkan tindakan bidang data. Ini adalah perubahan besar dalam kebijakan IAM yang ada yang didasarkan pada `connect` tindakan usang harus disesuaikan untuk menggunakan tindakan bidang data yang lebih terperinci. Lihat Jenis kebijakan IAM.

- Peningkatan ketersediaan instance pembaca. Sebelumnya, ketika instance penulis dimulai ulang, semua instance pembaca di cluster Neptunus juga otomatis dimulai ulang. Dimulai dengan rilis mesin 1.2.0.0, instance pembaca tetap aktif setelah penulis memulai ulang, yang meningkatkan ketersediaan pembaca. Instance pembaca dapat dimulai ulang secara terpisah untuk mengambil perubahan grup parameter. Lihat Mem-boot ulang instans DB di Amazon Neptunus.
- Menambahkan parameter cluster DB neptune_streams_expiry_days baru yang memungkinkan Anda mengatur jumlah hari rekaman streaming disimpan di server sebelum dihapus. Kisarannya adalah 1 hingga 90, dan defaultnya adalah 7.

Perbaikan dalam Rilis Mesin Ini

- Peningkatan kinerja serialisasi Gremlin untuk kueri. ByteCode
- Neptunus sekarang memproses predikat teks menggunakan mesin DFE, untuk meningkatkan kinerja.
- Neptunus sekarang memproses langkah-langkah `limit()` Gremlin menggunakan mesin DFE, termasuk batas traversal non-terminal dan anak.
- Mengubah penanganan DFE dari `union()` langkah Gremlin untuk bekerja dengan fitur baru lainnya, yang berarti bahwa node referensi muncul di profil kueri seperti yang diharapkan.
- Peningkatan kinerja hingga faktor 5 dari beberapa operasi gabungan yang mahal dalam DFE dengan memparalelkannya.
- Menambahkan dukungan `by()` modulasi `OrderGlobalStep order(global)` untuk mesin Gremlin DFE.
- Menambahkan tampilan nilai statis yang disuntikkan dalam menjelaskan detail untuk DFE.
- Peningkatan kinerja saat memangkas pola duplikat.
- Menambahkan dukungan pelestarian pesanan di mesin Gremlin DFE.
- Meningkatkan kinerja query Gremlin memiliki filter kosong, seperti ini:

```
g.V().hasId(P.within([]))
```

```
g.V().hasId([])
```

- Perpesanan kesalahan yang ditingkatkan ketika kueri SPARQL menggunakan nilai numerik yang terlalu besar untuk diwakili Neptunus secara internal.

- Peningkatan kinerja untuk menjatuhkan simpul dengan tepi terkait dengan mengurangi pencarian indeks saat aliran dinonaktifkan.
- Dukungan DFE yang diperluas ke lebih banyak varian `has()` langkah, khususnya untuk, `hasKey()``hasLabel()`, dan untuk berbagai predikat untuk di dalam. strings/URIs `has()` Ini memengaruhi kueri seperti berikut:

```
// hasKey() on properties
g.V().properties().hasKey("name")
g.V().properties().has(T.key, TextP.startingWith("a"))
g.E().properties().hasKey("weight")
g.E().properties().hasKey(TextP.containing("t"))

// hasLabel() on vertex properties
g.V().properties().hasLabel("name")

// range predicates on ID and Label fields
g.V().has(T.label, gt("person"))
g.E().has(T.id, lte("(an ID value)"))
```

- Menambahkan `join()`fungsi OpenCypher khusus Neptunus yang menggabungkan string dalam daftar menjadi satu string.
- Memperbarui kebijakan terkelola Neptunus untuk menyertakan izin akses data dan izin untuk database global baru. APIs

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug di mana permintaan HTTP tanpa tipe konten yang ditentukan akan secara otomatis gagal.
- Memperbaiki bug SPARQL di pengoptimal kueri yang mencegah penggunaan panggilan layanan di dalam kueri.
- Memperbaiki bug SPARQL di parser Turtle RDF di mana kombinasi tertentu dari data Unicode menyebabkan kegagalan.
- Memperbaiki bug SPARQL di mana kombinasi `GRAPH` dan `SELECT` klausa tertentu menghasilkan hasil kueri yang salah.
- Memperbaiki bug Gremlin yang menyebabkan masalah kebenaran untuk kueri yang menggunakan langkah filter apa pun dalam langkah gabungan, seperti berikut ini:

```
g.V("1").union(hasLabel("person"), out())
```

- Memperbaiki bug Gremlin di mana `count()` `both().simplePath()` akan menghasilkan dua kali lipat jumlah hasil aktual yang dikembalikan tanpa. `count()`
- Memperbaiki bug OpenCypher di mana pengecualian ketidakcocokan tanda tangan yang salah dihasilkan oleh server untuk permintaan Bolt ke cluster dengan otentikasi IAM diaktifkan.
- Memperbaiki bug OpenCypher di mana kueri menggunakan HTTP keep-alive bisa salah ditutup jika dikirimkan setelah permintaan gagal.
- Memperbaiki bug OpenCypher yang dapat menyebabkan kesalahan internal dilemparkan saat kueri yang mengembalikan nilai konstan dikirimkan.
- Memperbaiki bug dalam detail penjelasan sehingga subquery DFE `Time(ms)` sekarang dengan benar menjumlahkan waktu CPU operator dalam subquery DFE. Pertimbangkan kutipan berikut menjelaskan output sebagai contoh:

```
subQuery1
#######################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments                           #
 Mode # Units In # Units Out # Ratio # Time (ms) #
#######################################################################################
  ...
#######################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=...graph#336e.../graph_1 #
 -    # 1        # 1         # 1.00  # 0.38      #
#    #        #        #                       # coordinationTime(ms)=0.026        #
     #          #           #       #           #
#######################################################################################
  ...
subQuery=...graph#336e.../graph_1
#######################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments                           #
 Mode # Units In # Units Out # Ratio # Time (ms) #
#######################################################################################
# 0  # 1      # -      # DFESolutionInjection  # solutions=[?100 -> [-10^^<LONG>]] #
 -    # 0        # 1         # 0.00  # 0.04      #
#    #        #        #                       # outSchema=[?100]                  #
     #          #           #       #           #
#######################################################################################
# 1  # 3      # -      # DFERelationalJoin     # joinVars=[]                         #
 -    # 2        # 1         # 0.50  # 0.29      #
```

```
#################################################################################################
# 2  # 1      # -      # DFESolutionInjection  # outSchema=[]                                    #
 -    # 0        # 1         # 0.00  # 0.01      #
#################################################################################################
# 3  # -      # -      # DFEDrain              # -                                               #
 -    # 1        # 0         # 0.00  # 0.02      #
#################################################################################################
```

Waktu subQuery di kolom terakhir dari tabel bawah bertambah hingga 0,36 ms (`.04 + .29 + .01 + .02 = .36`. Ketika Anda menambahkan ke waktu koordinasi untuk subquery (`.36 + .026 = .386`), Anda mendapatkan hasil yang mendekati waktu untuk subQuery direkam di kolom terakhir dari tabel atas, yaitu `0.38` ms.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.0.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.0.0

Karena ini adalah rilis mesin utama, tidak ada peningkatan otomatis untuk itu.

Anda hanya dapat meningkatkan untuk merilis `1.2.0.0` secara manual, dari rilis patch terbaru dari rilis mesin 1.1.1.0. Rilis mesin sebelumnya harus terlebih dahulu ditingkatkan ke rilis terbaru `1.1.1.0` sebelum dapat ditingkatkan ke. `1.2.0.0`

Oleh karena itu, sebelum Anda mencoba meningkatkan ke rilis ini, harap konfirmasikan bahwa Anda sedang menjalankan rilis patch terbaru dari rilis`1.1.1.0`. Jika tidak, mulailah dengan memutakhirkan ke rilis patch terbaru dari. `1.1.1.0`

Sebelum memutakhirkan, Anda juga harus membuat ulang grup parameter cluster DB kustom yang telah Anda gunakan dengan versi sebelumnya, menggunakan keluarga grup parameter. `neptune1.2` Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

Jika Anda memutakhirkan terlebih dahulu untuk merilis `1.1.1.0` dan kemudian segera ke`1.2.0.0`, Anda mungkin mengalami kesalahan seperti berikut:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai (lihatMempertahankan Cluster DB Amazon Neptunus Anda).

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.0.0.R4 (2023-09-29)

Pada 2023-09-29, versi mesin 1.2.0.0.R4 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:
>
> - Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.
> - Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum

peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.

- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

- Menambahkan `enableInterContainerTrafficEncryption` parameter ke semua Neptunus APIs Neptunus, yang dapat Anda gunakan untuk mengaktifkan dan menonaktifkan enkripsi lalu lintas antar-kontainer dalam pelatihan atau pekerjaan penyetelan hiper-parameter.
- Untuk cluster DB tanpa server, ubah pengaturan kapasitas minimum menjadi 1.0 NCU, dan pengaturan maksimum valid terendah menjadi 2,5. NCUs Lihat Penskalaan kapasitas dalam cluster DB Neptunus Tanpa Server *(((before release, this change needs to be reflected in the serverless page too)))*.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug OpenCypher di mana update-and-return kueri tidak menangani`orderBy`,, `limit` atau dengan benar. `skip`

- Memperbaiki bug OpenCypher yang memungkinkan parameter yang terkandung dalam satu permintaan diganti oleh parameter yang terkandung dalam permintaan simultan lainnya.
- Memperbaiki bug OpenCypher dalam penanganan transaksi Bolt.
- Memperbaiki masalah kebenaran Gremlin untuk kueri DFE dengan `limit` sebagai traversal anak dari langkah-langkah non-serikat dengan kembali ke Tinkerpop. Misalnya, untuk kueri seperti ini:

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
 .select("a")
 .by(out()
 .limit(1))
```

- Memperbaiki bug Gremlin di mana kueri akan gagal karena berisi terlalu banyak TinkerPop langkah dan kemudian tidak akan dibersihkan.
- Memperbaiki bug Gremlin di mana tidak `order()` akan mengurutkan output string dengan benar ketika beberapa di antaranya berisi karakter spasi.
- Memperbaiki bug Gremlin di mana kebocoran transaksi dapat terjadi ketika kueri dikirimkan sebagai String dan berisi. `GroupCountStep`
- Memperbaiki bug Gremlin di mana kebocoran transaksi akan terjadi saat memeriksa titik akhir status kueri Gremlin untuk kueri dengan predikat di lintasan anak untuk langkah-langkah yang tidak diproses secara asli.
- Memperbaiki bug Gremlin tempat menambahkan Edge dan propertinya diikuti oleh `inV()` atau `outV()` menyebabkan file. `InternalFailureException`
- Memperbaiki masalah konkurensi di lapisan penyimpanan.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.0.0.R4, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.6`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.0.0.R4

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.2.0.0`.

Anda hanya dapat memutakhirkan untuk merilis `1.2.0.0` secara manual dari rilis patch terbaru dari rilis mesin 1.1.1.0. Rilis mesin sebelumnya harus terlebih dahulu ditingkatkan ke rilis terbaru `1.1.1.0` sebelum dapat ditingkatkan ke. `1.2.0.0`

Jika Anda memutakhirkan terlebih dahulu untuk merilis `1.1.1.0` dan kemudian segera ke`1.2.0.0`, Anda mungkin mengalami kesalahan seperti berikut:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
```

```
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.0.0.R3 (2022-12-15)

Pada 2022-12-15, versi mesin 1.2.0.0.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:
>
> - Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan

keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

- Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

  Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

  Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.
- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

- Peningkatan kinerja kueri OpenCypher yang melibatkan dan. `MERGE OPTIONAL MATCH`
- Meningkatkan kinerja query OpenCypher yang melibatkan `UNWIND` pada daftar peta nilai literal.
- Peningkatan kinerja kueri OpenCypher yang memiliki filter untuk. `IN id` Misalnya:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Peningkatan kinerja dan perbaikan kebenaran untuk berbagai operator Gremlin, termasuk`repeat`,,, dan. `coalesce` `store` `aggregate`

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug OpenCypher di mana kueri mengembalikan string,`"null"`, bukan nilai null di Bolt dan SPARQL-JSON.
- Memperbaiki bug OpenCypher sehingga dapat menafsirkan jenis parameter dengan benar ketika nilainya adalah daftar atau daftar peta.
- Memperbaiki bug log audit yang menyebabkan informasi yang tidak perlu dicatat dan bidang tertentu hilang dari log.
- Memperbaiki bug log audit di mana IAM ARN permintaan HTTP ke kluster DB berkemampuan IAM tidak direkam.
- Memperbaiki bug lookup-cache sehingga dapat membatasi memori tambahan yang digunakan untuk menulis ke cache.
- Memperbaiki bug lookup-cache yang melibatkan pengaturan mode hanya-baca untuk cache pencarian saat penulisan gagal.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.0.0.R3, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.0.0.R3

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.2.0.0`.

Anda hanya dapat memutakhirkan untuk merilis `1.2.0.0` secara manual dari rilis patch terbaru dari rilis mesin 1.1.1.0. Rilis mesin sebelumnya harus terlebih dahulu ditingkatkan ke rilis terbaru `1.1.1.0` sebelum dapat ditingkatkan ke. `1.2.0.0`

Jika Anda memutakhirkan terlebih dahulu untuk merilis `1.1.1.0` dan kemudian segera ke`1.2.0.0`, Anda mungkin mengalami kesalahan seperti berikut:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu

akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.2.0.0.R2 (2022-10-14)

Pada 2022-10-14, versi mesin 1.2.0.0.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Note**
>
> Jika memutakhirkan dari versi mesin lebih awal dari 1.2.0.0:
>
> - Engine release 1.2.0.0 memperkenalkan format baru untuk grup parameter kustom dan grup parameter cluster kustom. Akibatnya, jika Anda memutakhirkan dari versi engine lebih awal dari 1.2.0.0 ke engine versi 1.2.0.0 atau lebih tinggi, Anda harus membuat ulang semua grup parameter kustom yang ada dan grup parameter cluster kustom menggunakan keluarga grup parameter. `neptune1.2` Rilis sebelumnya menggunakan keluarga grup parameter`neptune1`, dan grup parameter tersebut tidak akan berfungsi dengan rilis 1.2.0.0 ke atas. Untuk informasi selengkapnya, lihat Grup parameter Amazon Neptunus.

- Engine release 1.2.0.0 juga memperkenalkan format baru untuk membatalkan log. Akibatnya, setiap log pembatalan yang dibuat oleh versi mesin sebelumnya harus dibersihkan dan `UndoLogsListSize` CloudWatch metrik harus jatuh ke nol sebelum peningkatan apa pun dari versi yang lebih awal dari 1.2.0.0 dapat dimulai. Jika ada terlalu banyak catatan log undo (200.000 atau lebih) saat Anda mencoba memulai pembaruan, upaya pemutakhiran dapat habis sementara menunggu pembersihan log pembatalan selesai.

  Anda dapat mempercepat tingkat pembersihan dengan memutakhirkan instance penulis cluster, di mana pembersihan terjadi. Melakukan itu sebelum mencoba memutakhirkan dapat menurunkan jumlah log batal sebelum Anda mulai. Meningkatkan ukuran penulis ke jenis instans 24XL dapat meningkatkan tingkat pembersihan Anda menjadi lebih dari satu juta catatan per jam.

  Jika `UndoLogsListSize` CloudWatch metriknya sangat besar, membuka kasus dukungan dapat membantu Anda mengeksplorasi strategi tambahan untuk menurunkannya.
- Akhirnya, ada perubahan besar dalam rilis 1.2.0.0 yang mempengaruhi kode sebelumnya yang menggunakan protokol Bolt dengan otentikasi IAM. Dimulai dengan rilis 1.2.0.0, Bolt membutuhkan jalur sumber daya untuk penandatanganan IAM. Di Java, menyetel jalur sumber daya mungkin terlihat seperti ini:`request.setResourcePath("/openCypher"));`. Dalam bahasa lain, `/openCypher` dapat ditambahkan ke URI endpoint. Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

- Peningkatan kinerja kueri Gremlin`order-by`. Kueri Gremlin dengan `order-by` di akhir `NeptuneGraphQueryStep` sekarang menggunakan ukuran potongan yang lebih besar untuk kinerja yang lebih baik. Ini tidak berlaku untuk `order-by` node internal (non-root) dari rencana kueri.
- Peningkatan kinerja kueri pembaruan Gremlin. Simpul dan tepi sekarang harus dikunci terhadap penghapusan sambil menambahkan tepi atau properti. Perubahan ini menghilangkan kunci duplikat dalam transaksi, yang meningkatkan kinerja.
- Peningkatan kinerja query Gremlin yang menggunakan `dedup()` bagian dalam `repeat()` subquery dengan menekan `dedup` ke bawah ke layer eksekusi asli.

- Ditambahkan petunjuk `Neptune#cardinalityEstimates` query Gremlin. Ketika diatur ke`false`, ini menonaktifkan perkiraan kardinalitas.
- Menambahkan pesan kesalahan yang mudah digunakan untuk kesalahan otentikasi IAM. Pesan-pesan ini sekarang menunjukkan pengguna IAM Anda atau peran ARN, ARN sumber daya, dan daftar tindakan yang tidak sah untuk permintaan tersebut. Daftar tindakan yang tidak sah membantu Anda melihat apa yang mungkin hilang atau ditolak secara eksplisit dalam kebijakan IAM yang Anda gunakan.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug kebenaran Gremlin yang melibatkan `WherePredicateStep` terjemahan, di mana mesin kueri Neptunus menghasilkan hasil yang salah untuk kueri yang digunakan dan variasinya. `where(P.neq('x'))`
- Memperbaiki bug Gremlin di mana menggunakan `PartitionStrategy` setelah memutakhirkan ke TinkerPop 3.5 secara tidak benar menghasilkan kesalahan dengan pesan, "PartitionStrategy tidak berfungsi dengan Traversals anonim," yang mencegah traversal dieksekusi.
- Memperbaiki berbagai bug Gremlin yang terkait dengan `joinTime` gabungan akhir dan statistik di dalam subkelompok. `Project.ASK`
- Memperbaiki bug OpenCypher di `MERGE` klausa yang dalam beberapa kasus menyebabkan duplikat node dan pembuatan edge.
- Memperbaiki bug transaksi di mana sesi dapat menyisipkan data grafik dan komit bahkan ketika sisipan kamus bersamaan yang sesuai dibatalkan.
- Memperbaiki bug pemuat massal yang menyebabkan regresi kinerja di bawah beban penyisipan berat.
- Memperbaiki bug SPARQL dalam penanganan kueri yang berisi `(NOT) EXISTS` dalam `OPTIONAL` klausa, di mana dalam beberapa kasus hasil kueri hilang.
- Memperbaiki bug di mana driver dapat tampak hang dalam kasus di mana permintaan dibatalkan karena batas waktu sebelum dimulainya evaluasi. Dimungkinkan untuk masuk ke status ini jika semua utas pemrosesan kueri di server dikonsumsi saat batas waktu terjadi pada item dalam antrian permintaan. Karena batas waktu dari antrian permintaan tidak segera mengirim pesan, tanggapan muncul kepada klien untuk tetap tertunda.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.2.0.0.R2, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.2.0.0.R2

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.2.0.0`.

Anda hanya dapat memutakhirkan untuk merilis `1.2.0.0` secara manual dari rilis patch terbaru dari rilis mesin 1.1.1.0. Rilis mesin sebelumnya harus terlebih dahulu ditingkatkan ke rilis terbaru `1.1.1.0` sebelum dapat ditingkatkan ke. `1.2.0.0`

Jika Anda memutakhirkan terlebih dahulu untuk merilis `1.1.1.0` dan kemudian segera ke`1.2.0.0`, Anda mungkin mengalami kesalahan seperti berikut:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.1.1.0 (2022-04-19)

Pada 2022-04-19, engine versi 1.1.1.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Important

Memutakhirkan ke rilis mesin ini dari versi sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.

Agar berhasil menyelesaikan peningkatan, setiap subnet di setiap zona ketersediaan (AZ) harus memiliki setidaknya satu alamat IP yang tersedia per instance Neptunus. Misalnya, jika ada satu instance penulis dan dua instance pembaca di subnet 1, dan dua instance pembaca di subnet 2, subnet 1 harus memiliki setidaknya 3 alamat IP gratis dan subnet 2 harus memiliki setidaknya 2 alamat IP gratis sebelum memulai upgrade.

Pada awal peningkatan, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.

Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama beberapa menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.

Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`

- `Finished applying off-line patches to DB instance`
- `DB instance restarted`

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis pemeliharaan: 1.1.1.0.R2 (2022-05-16)
- Rilis: 1.1.1.0.R3 (2022-06-07)
- Rilis: 1.1.1.0.R4 (2022-06-23)
- Rilis: 1.1.1.0.R5 (2022-07-21)
- Rilis: 1.1.1.0.R6 (2022-09-23)
- Rilis: 1.1.1.0.R7 (2023-01-23)

Fitur Baru dalam Rilis Mesin Ini

- Bahasa query OpenCypher sekarang umumnya tersedia untuk penggunaan produksi.

> **Warning**
>
> Ada perubahan besar dalam rilis ini untuk kode yang menggunakan OpenCypher dengan otentikasi IAM. Dalam pratinjau Neptunus untuk OpenCypher, string host dalam tanda tangan IAM menyertakan protokol, seperti, seperti ini: `bolt://`
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> Dimulai dengan rilis mesin ini, protokol harus dihilangkan:
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> Lihat Menggunakan protokol Bolt sebagai contoh.

- Menambahkan dukungan untuk TinkerPop `3.5.2`. Di antara perubahan dalam rilis ini adalah dukungan untuk transaksi jarak jauh dan dukungan bytecode untuk sesi (menggunakan `g.tx`), dan penambahan `datetime()` fungsi ke bahasa Gremlin.

> **Warning**
>
> Ada beberapa perubahan besar yang diperkenalkan di TinkerPop 3.5.0, 3.5.1, dan 3.5.2 yang dapat memengaruhi kode Gremlin Anda. Misalnya, menggunakan traversal yang dihasilkan oleh a GraphTraversalSource sebagai anak-anak seperti ini tidak akan berfungsi lagi:. `g.V().union(identity(), g.V())`
> Sekarang sebagai gantinya, gunakan traversal anonim seperti ini:`g.V().union(identity(), __.V())`.

- Menambahkan dukungan untuk kunci kondisi AWS global yang dapat Anda gunakan dalam kebijakan akses data IAM yang mengontrol akses ke data yang disimpan di Neptunus dan cluster DB Neptunus.
- Mesin kueri Neptunus DFE sekarang umumnya tersedia untuk penggunaan produksi dengan bahasa kueri OpenCypher, tetapi belum untuk kueri Gremlin dan SPARQL. Anda sekarang mengaktifkannya menggunakan parameter neptune_dfe_query_engine instance-nya sendiri daripada parameter lab-mode.

Perbaikan dalam Rilis Mesin Ini

- Menambahkan fitur baru ke OpenCypher seperti dukungan kueri berparameter, caching pohon sintaks abstrak (AST) untuk kueri berparameter, peningkatan jalur panjang variabel (VLP), serta operator dan klausa baru. Lihat Kepatuhan spesifikasi OpenCypher di Amazon Neptunus untuk tingkat dukungan bahasa saat ini.
- Membuat peningkatan kinerja yang signifikan pada OpenCypher untuk beban kerja baca dan tulis yang sederhana, menghasilkan throughput yang lebih tinggi jika dibandingkan dengan Release 1.1.0.0.
- Menghapus keterbatasan dua arah dan kedalaman OpenCypher yang menangani jalur panjang variabel.
- Dukungan lengkap di mesin DFE untuk Gremlin `within` dan `without` predikat, termasuk kasus di mana mereka digabungkan dengan operator predikat lainnya. Misalnya:

```
g.V().has('age', within(12, 15, 18).or(gt(30)))
```

- Dukungan yang diperluas di mesin DFE untuk `order` langkah Gremlin ketika ruang lingkup global (yaitu, tidak`order(local)`), dan ketika `by()` modulator tidak digunakan. Misalnya, kueri ini sekarang memiliki dukungan DFE:

```
g.V().values("age").order()
```

- Menambahkan `isLastOp` bidang ke format respons perubahan log aliran Neptunus, untuk menunjukkan bahwa catatan adalah operasi terakhir dalam transaksinya.
- Secara signifikan meningkatkan kinerja pencatatan audit dan mengurangi latensi saat pencatatan audit diaktifkan.
- Dikonversi WebSocket bytecode Gremlin dan kueri HTTP menjadi format yang dapat dibaca pengguna di log audit. Kueri sekarang dapat langsung disalin dari log audit untuk dieksekusi di notebook Neptunus dan di tempat lain. Perhatikan bahwa perubahan pada format log audit saat ini merupakan perubahan yang melanggar.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin langka di mana tidak ada hasil yang dikembalikan saat menggunakan bersarang `filter()` dan `count()` langkah-langkah dalam kombinasi, seperti dalam kueri berikut:

```
g.V("1").filter(out("knows")
        .filter(in("knows")
        .hasId("notExists"))
        .count())
```

- Memperbaiki bug Gremlin di mana kesalahan dikembalikan saat menggunakan simpul yang disimpan oleh langkah agregat di salah satu `to()` atau `from()` traversal dalam hubungannya dengan langkah. `addE` Contoh kueri semacam itu adalah:

```
g.V("id").aggregate("v").out().addE().to(select("v").unfold()))
```

- Memperbaiki bug Gremlin di mana `not` langkahnya gagal dalam kasus tepi saat menggunakan mesin DFE. Misalnya:

```
g.V().not(V())
```

- Memperbaiki bug Gremlin di mana `sideEffect` nilai tidak tersedia di dalam `to()` dan `from()` melintasi.

- Memperbaiki bug yang terkadang menyebabkan reset cepat memicu failover instance.
- Memperbaiki bug pemuat massal di mana transaksi yang gagal tidak akan ditutup sebelum memulai pekerjaan pemuatan berikutnya.
- Memperbaiki bug loader massal di mana kondisi memori rendah dapat menyebabkan crash pada sistem.
- Menambahkan percobaan lagi untuk memperbaiki bug pemuat massal di mana loader tidak menunggu cukup lama hingga kredensyal IAM tersedia setelah failover.
- Memperbaiki bug di mana cache kredensyal internal tidak dihapus dengan benar untuk titik akhir non-kueri seperti titik akhir. `status`
- Memperbaiki bug aliran untuk memastikan bahwa nomor urutan komit aliran diurutkan dengan benar.
- Memperbaiki bug di mana koneksi yang berjalan lama dihentikan lebih cepat dari sepuluh hari pada kluster yang mendukung IAM.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.1.1.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.1.1.0

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini. Perhatikan bahwa versi sebelum mesin versi utama (1.1.0.0) akan membutuhkan waktu lebih lama untuk meningkatkan ke rilis ini.

Anda tidak akan secara otomatis ditingkatkan ke rilis ini.

Peningkatan ke Rilis Ini

Important

Memutakhirkan ke rilis mesin ini dari versi apa pun sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.

Pada awal peningkatan, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.

Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama sekitar 6 menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.

Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine neptune \
    --engine-version 1.1.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine neptune ^
    --engine-version 1.1.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.1.1.0.R7 (2023-01-23)

Pada 2023-01-23, versi mesin 1.1.1.0.R7 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Important

Memutakhirkan ke rilis mesin ini dari versi sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.

Untuk menyelesaikan peningkatan dengan sukses, setiap subnet di setiap zona ketersediaan (AZ) harus memiliki setidaknya satu alamat IP yang tersedia per instance Neptunus. Misalnya, jika ada satu instance penulis dan dua instance pembaca di subnet 1, dan dua instance pembaca di subnet 2, subnet 1 harus memiliki setidaknya 3 alamat IP gratis dan subnet 2 harus memiliki setidaknya 2 alamat IP gratis sebelum memulai upgrade.

Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.

Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama beberapa menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.

Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`

- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Perbaikan dalam Rilis Mesin Ini

- Peningkatan kinerja kueri OpenCypher yang melibatkan dan. `MERGE` `OPTIONAL MATCH`
- Peningkatan kinerja query OpenCypher yang melibatkan `UNWIND` daftar peta nilai literal.
- Peningkatan kinerja kueri OpenCypher yang memiliki filter untuk. `IN` `id` Misalnya:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Peningkatan kinerja dan perbaikan kebenaran untuk berbagai operator Gremlin, termasuk`repeat`,,, dan. `coalesce` `store` `aggregate`

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug OpenCypher di mana permintaan menggunakan HTTP keep-alive bisa salah ditutup jika dikirimkan setelah permintaan gagal.
- Memperbaiki bug OpenCypher di mana jenis parameter tidak selalu ditafsirkan dengan benar untuk daftar atau daftar peta.
- Memperbaiki bug OpenCypher di mana kueri mengembalikan string,`"null"`, bukan nilai null di Bolt dan SPARQL-JSON.
- Memperbaiki kode kesalahan OpenCypher dan pesan kesalahan untuk kegagalan dan kesalahan batas waktu kueri. out-of-memory
- Memperbaiki bug Gremlin yang menyebabkan `valueMap()` tidak dioptimalkan di bawah `by()` traversal di mesin DFE.
- Memperbaiki masalah dengan logika detektor kebuntuan yang terkadang membuat mesin tidak responsif.
- Memperbaiki bug log audit yang menyebabkan informasi yang tidak perlu dicatat dan bidang tertentu hilang dari log.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.1.1.0.R7, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.3`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.1.1.0.R7

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi mesin`1.1.1.0`.

Meningkatkan ke Rilis Ini

Important

Memutakhirkan ke rilis mesin ini dari versi apa pun sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.

Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.

Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama sekitar 6 menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.

Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:

    - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
    - `Database cluster major version has been upgraded`
  - Pesan acara per instance:
    - `Applying off-line patches to DB instance`
    - `DB instance shutdown`
    - `Finished applying off-line patches to DB instance`
    - `DB instance restarted`

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.1.1.0.R6 (2022-09-23)

Pada 2022-09-23, versi mesin 1.1.1.0.R6 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Important

Memutakhirkan ke rilis mesin ini dari versi sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.

Agar berhasil menyelesaikan peningkatan, setiap subnet di setiap zona ketersediaan (AZ) harus memiliki setidaknya satu alamat IP yang tersedia per instance Neptunus. Misalnya, jika ada satu instance penulis dan dua instance pembaca di subnet 1, dan dua instance pembaca di subnet 2, subnet 1 harus memiliki setidaknya 3 alamat IP gratis dan subnet 2 harus memiliki setidaknya 2 alamat IP gratis sebelum memulai upgrade.

Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.

Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama beberapa menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.

Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`

- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> Ada perubahan besar dalam rilis ini untuk kode yang menggunakan OpenCypher dengan otentikasi IAM. Hingga saat ini, string host dalam tanda tangan IAM menyertakan protokol, sepertibolt://, seperti ini:
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> Dimulai dengan pelepasan mesin1.1.1.0, protokol harus dihilangkan:
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

- Peningkatan kinerja kueri Gremlinorder-by. Kueri Gremlin dengan order-by di akhir NeptuneGraphQueryStep sekarang menggunakan ukuran potongan yang lebih besar untuk kinerja yang lebih baik. Ini tidak berlaku untuk order-by node internal (non-root) dari rencana kueri.
- Peningkatan kinerja kueri pembaruan Gremlin. Simpul dan tepi sekarang harus dikunci terhadap penghapusan sambil menambahkan tepi atau properti. Perubahan ini menghilangkan kunci duplikat dalam transaksi, yang meningkatkan kinerja.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug OpenCypher di MERGE klausa yang dalam beberapa kasus menyebabkan duplikat node dan pembuatan edge.

- Memperbaiki bug dalam penanganan kueri SPARQL yang berisi `(NOT) EXISTS` dalam `OPTIONAL` klausa, di mana dalam beberapa kasus hasil kueri akan hilang.
- Memperbaiki bug yang menunda restart server saat beban massal sedang berlangsung.
- Memperbaiki bug di mana traversal bi-directional pola panjang variabel OpenCypher dengan filter pada properti relasi akan mengakibatkan kesalahan. Contoh pola panjang variabel seperti itu adalah. `(n)-[r*1..2]->(m)`
- Memperbaiki bug yang terkait dengan cara data cache dikirim kembali ke klien, yang dalam beberapa kasus mengakibatkan latensi panjang yang tak terduga.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.1.1.0.R6, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.1.1.0.R6

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi mesin`1.1.1.0`.

Meningkatkan ke Rilis Ini

Important

Memutakhirkan ke rilis mesin ini dari versi apa pun sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.
Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat

menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.
Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama sekitar 6 menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.
Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
```

```
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.1.1.0.R5 (2022-07-21)

Pada 2022-07-21, versi mesin 1.1.1.0.R5 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Important

Memutakhirkan ke rilis mesin ini dari versi sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.

Agar berhasil menyelesaikan peningkatan, setiap subnet di setiap zona ketersediaan (AZ) harus memiliki setidaknya satu alamat IP yang tersedia per instance Neptunus. Misalnya, jika ada satu instance penulis dan dua instance pembaca di subnet 1, dan dua instance pembaca di subnet 2, subnet 1 harus memiliki setidaknya 3 alamat IP gratis dan subnet 2 harus memiliki setidaknya 2 alamat IP gratis sebelum memulai upgrade.

Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.

Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama beberapa menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.
Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Ada perubahan besar dalam rilis ini untuk kode yang menggunakan OpenCypher dengan otentikasi IAM. Hingga saat ini, string host dalam tanda tangan IAM menyertakan protokol, sepertibolt://, seperti ini:

```
"Host":"bolt://(host URL):(port)"
```

Dimulai dengan pelepasan mesin`1.1.1.0`, protokol harus dihilangkan:

```
"Host":"(host URL):(port)"
```

Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

- Melakukan perbaikan untuk mendukung deteksi kebuntuan.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug yang mencegah shutdown bersih cluster DB dalam kondisi tertentu.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.1.1.0.R5, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.1.1.0.R5

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi mesin`1.1.1.0`.

Meningkatkan ke Rilis Ini

Important

Memutakhirkan ke rilis mesin ini dari versi apa pun sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.

Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.

Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan

tersedia pada saat ini selama sekitar 6 menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.
Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.1.1.0.R4 (2022-06-23)

Pada 2022-06-23, versi mesin 1.1.1.0.R4 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Important

Memutakhirkan ke rilis mesin ini dari versi sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.

Untuk menyelesaikan peningkatan dengan sukses, setiap subnet di setiap zona ketersediaan (AZ) harus memiliki setidaknya satu alamat IP yang tersedia per instance Neptunus. Misalnya, jika ada satu instance penulis dan dua instance pembaca di subnet 1, dan dua instance pembaca di subnet 2, subnet 1 harus memiliki setidaknya 3 alamat IP gratis dan subnet 2 harus memiliki setidaknya 2 alamat IP gratis sebelum memulai upgrade.

Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.

Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama beberapa menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.

Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Ada perubahan besar dalam rilis ini untuk kode yang menggunakan OpenCypher dengan otentikasi IAM. Hingga saat ini, string host dalam tanda tangan IAM menyertakan protokol, sepertibolt://, seperti ini:

```
"Host":"bolt://(host URL):(port)"
```

Dimulai dengan pelepasan mesin1.1.1.0, protokol harus dihilangkan:

```
"Host":"(host URL):(port)"
```

Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

- Diperbarui konfigurasi x2g instance untuk jenis contoh.
- Peningkatan kinerja tetes vertex.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin di mana solusi tidak mempertahankan urutan stabil untuk kueri yang dipanggil beberapa kali atau di beberapa pembaca untuk jenis gabungan ASK tertentu.

- Juga, mempersempit ruang lingkup perubahan dalam rilis sebelumnya yang menyebabkan regresi kinerja untuk jenis ASK tertentu bergabung di Gremlin.
- Memperbaiki bug Gremlin pada `union()` langkah yang terjadi ketika ada input tepi dan traversal ke simpul dalam traversal anak.
- Memperbaiki bug profil Gremlin di mana beberapa langkah dilaporkan tidak dioptimalkan saat sebenarnya.
- Memperbaiki bug SPARQL di mana variabel yang digunakan di dalam `FILTER` ekspresi yang disarangkan ke dalam `UNION` klausa mendapatkan informasi pelingkupan yang tidak valid.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.1.1.0.R4, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.1.1.0.R4

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi mesin`1.1.1.0`.

Meningkatkan ke Rilis Ini

Important

Memutakhirkan ke rilis mesin ini dari versi apa pun sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.
Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat

menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.
Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama sekitar 6 menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.
Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
```

```
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.1.1.0.R3 (2022-06-07)

Pada 2022-06-07, versi mesin 1.1.1.0.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Important

Memutakhirkan ke rilis mesin ini dari versi sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.

Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.

Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama beberapa menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.

Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*(autogenerated snapshot ID)*`]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Ada perubahan besar dalam rilis ini untuk kode yang menggunakan OpenCypher dengan otentikasi IAM. Hingga saat ini, string host dalam tanda tangan IAM menyertakan protokol, sepertibolt://, seperti ini:

```
"Host":"bolt://(host URL):(port)"
```

Dimulai dengan pelepasan mesin`1.1.1.0`, protokol harus dihilangkan:

```
"Host":"(host URL):(port)"
```

Lihat Menggunakan protokol Bolt sebagai contoh.

Perbaikan dalam Rilis Mesin Ini

- Menambahkan dukungan untuk jenis `x2g` instans bertenaga Graviton2, yang dioptimalkan untuk beban kerja intensif memori. Ini awalnya hanya tersedia dalam empat Wilayah AWS:
  - US East (N. Virginia) (`us-east-1`)
  - US East (Ohio) (`us-east-2`)
  - US West (Oregon) (`us-west-2`)
  - Europe (Ireland) (`eu-west-1`)

Lihat halaman harga Neptunus untuk informasi lebih lanjut.

- Peningkatan kinerja langkah-langkah Gremlin di mana beberapa lintasan tepi atau simpul, pencarian properti, atau pencarian label terlibat.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin dalam pemrosesan `otherV()` langkah di dalam traversal anak.
- Memperbaiki bug Gremlin dalam kueri dengan hanya `union` memiliki langkah-langkah filter sebagai anak-anak. Misalnya:

```
g.V().union(has("name"), out("knows")).out()
```

- Memperbaiki bug SPARQL di mana variabel yang digunakan di dalam `FILTER` ekspresi yang disarangkan ke dalam `UNION` klausa mendapatkan informasi pelingkupan yang tidak valid.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.1.1.0.R3, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.1.1.0.R3

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi mesin`1.1.1.0`.

Meningkatkan ke Rilis Ini

Important

Memutakhirkan ke rilis mesin ini dari versi apa pun sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan

penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.
Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.
Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama sekitar 6 menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.
Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
```

```
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu

akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Rilis pemeliharaan Amazon Neptunus, versi 1.1.1.0.R2 (2022-05-16)

Pada 2022-05-16, versi mesin 1.1.1.0.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Important**
>
> Memutakhirkan ke rilis mesin ini dari versi sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.
> Untuk menyelesaikan peningkatan dengan sukses, setiap subnet di setiap zona ketersediaan (AZ) harus memiliki setidaknya satu alamat IP yang tersedia per instance Neptunus.
> Misalnya, jika ada satu instance penulis dan dua instance pembaca di subnet 1, dan dua

instance pembaca di subnet 2, subnet 1 harus memiliki setidaknya 3 alamat IP gratis dan subnet 2 harus memiliki setidaknya 2 alamat IP gratis sebelum memulai upgrade.
Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.
Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama beberapa menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.
Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Ada perubahan besar dalam rilis ini untuk kode yang menggunakan OpenCypher dengan otentikasi IAM. Hingga saat ini, string host dalam tanda tangan IAM menyertakan protokol, sepertibolt://, seperti ini:

```
"Host":"bolt://(host URL):(port)"
```

Dimulai dengan pelepasan mesin`1.1.1.0`, protokol harus dihilangkan:

```
"Host":"(host URL):(port)"
```

Lihat Menggunakan protokol Bolt sebagai contoh.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.1.1.0.R2, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi paling awal Gremlin didukung: `3.5.2`
- Gremlin versi terbaru didukung: `3.5.4`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan jalur ke rilis mesin 1.1.1.0.R2

Cluster Anda akan ditingkatkan ke rilis patch pemeliharaan ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi `1.1.1.0` engine.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Important

Memutakhirkan ke rilis mesin ini dari versi apa pun sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.

Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.

Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama sekitar 6 menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.
Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.1.0.0 (2021-11-19)

Pada 2021-11-19, versi mesin 1.1.0.0 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Important

Memutakhirkan ke rilis mesin ini dari versi sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.

Agar berhasil menyelesaikan peningkatan, setiap subnet di setiap zona ketersediaan (AZ) harus memiliki setidaknya satu alamat IP yang tersedia per instance Neptunus. Misalnya, jika ada satu instance penulis dan dua instance pembaca di subnet 1, dan dua instance pembaca di subnet 2, subnet 1 harus memiliki setidaknya 3 alamat IP gratis dan subnet 2 harus memiliki setidaknya 2 alamat IP gratis sebelum memulai upgrade.

Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.

Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan

tersedia pada saat ini selama beberapa menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.
Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> Dimulai dengan rilis mesin ini, Neptunus tidak lagi R4 mendukung jenis instans. Jika Anda menggunakan R4 instance di cluster DB Anda, Anda harus menggantinya secara manual dengan jenis instans yang berbeda sebelum memutakhirkan ke rilis ini. Jika contoh penulis Anda adalahR4, ikuti instruksi ini untuk memindahkannya.

Rilis patch berikutnya untuk rilis ini

- Rilis pemeliharaan: 1.1.0.0.R2 (2022-05-16)
- Rilis pemeliharaan: 1.1.0.0.R3 (2022-12-23)

Fitur Baru dalam Rilis Mesin Ini

- Memperkenalkan instans R6g database tujuan umum T4g dan dioptimalkan memori yang didukung oleh prosesor Graviton2.AWS Instans berbasis Graviton2 memberikan hasil yang jauh lebih baik price/performance daripada instans berbasis x86 generasi saat ini yang sebanding untuk berbagai beban kerja. Aplikasi bekerja seperti biasa pada jenis instans baru ini, dan tidak perlu mem-port kode aplikasi saat Anda memutakhirkannya.

Untuk informasi selengkapnya tentang harga dan ketersediaan regional, lihat halaman harga Amazon Neptunus.

- Memperkenalkan model kustom di Neptunus ML.
- Menambahkan dukungan untuk kueri inferensi SPARQL di Neptunus ML.
- Menambahkan titik akhir aliran baru untuk data grafik properti, yaitu:

```
https://Neptune-DNS:8182/propertygraph/stream
```

Format output dari endpoint ini, bernama`PG_JSON`, persis sama dengan `GREMLIN_JSON` format output oleh yang lama`gremlin/stream`.

`propertygraph/stream`Endpoint baru memperluas dukungan aliran Neptunus ke OpenCypher dan menggantikan titik akhir dengan format output yang terkait. `gremlin/stream` `GREMLIN_JSON`

Perbaikan dalam Rilis Mesin Ini

- Membuat perbaikan pada aliran Neptunus:
  - Menambahkan `commitTimestamp` bidang ke `records` objek Neptunus mengalirkan format respons log perubahan, untuk memberikan stempel waktu untuk setiap catatan dalam aliran log perubahan.
  - Menambahkan `LATEST` nilai ke `iteratorType` parameter, memungkinkan Anda untuk mengambil eventiD valid terakhir dari aliran. Lihat Memanggil Streams API.
- Menambahkan dukungan untuk mendapatkan skor kepercayaan inferensi dalam klasifikasi node Gremlin dan kueri regresi.
- Ditambahkan dukungan untuk `OPTIONAL MATCH` klausa di OpenCypher.
- Ditambahkan dukungan untuk `MERGE` klausa di OpenCypher.
- Ditambahkan dukungan untuk menggunakan `ORDER BY` dalam `WITH` klausa di OpenCypher.
- Menambahkan dukungan untuk pemahaman pola di OpenCypher, dan dukungan yang diperluas untuk ekspresi pola di luar pemeriksaan keberadaan.
- Dukungan yang diperluas untuk `DELETE DETACH` klausa `DELETE` dan di OpenCypher, sehingga sekarang dapat digunakan dengan klausa pembaruan lainnya.

- Dukungan diperpanjang untuk `CREATE` dan `UPDATE` klausa yang digunakan `RETURN` di OpenCypher.
- Menambahkan dukungan di mesin DFE untuk Gremlin`limit`,`range`, dan langkah-langkah. `skip`
- Peningkatan eksekusi kueri di mesin DFE ketika tidak `profile` ada `explain` atau diminta.
- Peningkatan eksekusi kueri di mesin DFE untuk `value` ekspresi.
- Meningkatkan sejumlah pola penyisipan bersyarat Gremlin yang dirantai untuk menghindari pengecualian modifikasi bersamaan dan memungkinkan rantai pola kueri seperti ini:
  - Penyisipan simpul bersyarat dengan ID, seperti:

```
g.V(ID).fold().coalesce(unfold(), g.addV("L1").property(id,ID))
```

  - Penyisipan simpul bersyarat dengan beberapa label, seperti:

```
g.V(ID).fold().coalesce(unfold(), g.addV("L1::L2").property(id,ID))
```

  - Penyisipan tepi bersyarat oleh ID, seperti:

```
g.E(ID).fold().coalesce(unfold(), V(from).addE(label).to(V(to)).property(id, ID))
```

  - Penyisipan tepi bersyarat dengan beberapa label, seperti:

```
g.E(ID).fold().coalesce(unfold(),
 g.addE(label).from(V(from)).to(V(to)).property(id, ID))
```

  - Penyisipan bersyarat diikuti oleh kueri, seperti:

```
g.V(ID).fold().coalesce(unfold(),
 g.addV("L1").property(id,ID)).project("myvalues").by(valueMap())
```

  - Penyisipan bersyarat dengan properti tambahan, seperti:

```
g.V(ID).fold().coalesce(unfold(),
 g.addV("L1").property(id,ID).property("name","pumba"))
```

Perbaikan Cacat dalam Rilis Mesin Ini

- Menonaktifkan fitur statistik pada jenis `T3.medium` instance, yang tidak dapat mendukungnya.
- Memperbaiki bug SPARQL `explain` dengan `IN` fungsi yang mengambil nilai non-konstan.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.1.0.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.11`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.1.0.0

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Anda tidak akan secara otomatis ditingkatkan ke rilis ini.

Peningkatan ke Rilis Ini

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Alih-alih`--apply-immediately`, Anda dapat menentukan`--no-apply-immediately`. Untuk melakukan upgrade versi utama, allow-major-version-upgrade parameter diperlukan. Juga, pastikan untuk menyertakan versi mesin atau mesin Anda dapat ditingkatkan ke versi yang berbeda.

Jika klaster Anda menggunakan grup parameter cluster kustom, pastikan untuk menyertakan paramater ini untuk menentukannya:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Demikian pula, jika ada instance di cluster yang menggunakan grup parameter DB kustom, pastikan untuk menyertakan parameter ini untuk menentukannya:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Rilis pemeliharaan Amazon Neptunus, versi 1.1.0.0.R3 (2022-12-23)

Pada 2022-12-23, versi mesin 1.1.0.0.R3 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

> **Important**
>
> Memutakhirkan ke rilis mesin ini dari versi sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.
> Untuk menyelesaikan peningkatan dengan sukses, setiap subnet di setiap zona ketersediaan (AZ) harus memiliki setidaknya satu alamat IP yang tersedia per instance Neptunus.
> Misalnya, jika ada satu instance penulis dan dua instance pembaca di subnet 1, dan dua instance pembaca di subnet 2, subnet 1 harus memiliki setidaknya 3 alamat IP gratis dan subnet 2 harus memiliki setidaknya 2 alamat IP gratis sebelum memulai upgrade.
> Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi

cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.
Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama beberapa menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.
Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Perbaikan dalam Rilis Mesin Ini

- Peningkatan kinerja dan perbaikan kebenaran untuk berbagai operator Gremlin, termasuk`repeat`,,, dan. `coalesce` `store` `aggregate`

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki masalah lonjakan CPU.
- Memperbaiki bug OpenCypher di mana kueri mengembalikan string,`"null"`, bukan nilai null di Bolt dan SPARQL-JSON.
- Memperbaiki bug log audit yang menyebabkan informasi yang tidak perlu dicatat dan bidang tertentu hilang dari log.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.1.0.0.R3, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.11`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.1.0.0.R3

Cluster Anda akan ditingkatkan ke rilis patch pemeliharaan ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi `1.1.0.0` engine.

Important

Memutakhirkan ke rilis mesin ini dari versi apa pun sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.

Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.

Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama sekitar 6 menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.

Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`

- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> Dimulai dengan rilis mesin ini, Neptunus tidak lagi R4 mendukung jenis instans. Jika Anda menggunakan R4 instance di cluster DB Anda, Anda harus menggantinya secara manual dengan jenis instans yang berbeda sebelum memutakhirkan ke rilis ini. Jika contoh penulis Anda adalahR4, ikuti instruksi ini untuk memindahkannya.

Peningkatan ke Rilis Ini

Amazon Neptunus 1.1.0.0.R3 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Rilis pemeliharaan Amazon Neptunus, versi 1.1.0.0.R2 (2022-05-16)

Pada 2022-05-16, versi mesin 1.1.0.0.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Important

Memutakhirkan ke rilis mesin ini dari versi sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.

Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.

Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama beberapa menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.

Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:

    - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
    - `Database cluster major version has been upgraded`
  - Pesan acara per instance:
    - `Applying off-line patches to DB instance`
    - `DB instance shutdown`
    - `Finished applying off-line patches to DB instance`
    - `DB instance restarted`

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug di mana cache kredenal internal tidak dihapus dengan benar untuk titik akhir non-kueri seperti titik akhir status.
- Memperbaiki bug yang menyebabkan kelambatan replikasi meningkat setelah peningkatan mesin.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.1.0.0.R2, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.11`
- Versi OpenCypher: `Neptune-9.0.20190305-1.0`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.1.0.0.R2

Cluster Anda akan ditingkatkan ke rilis patch pemeliharaan ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi `1.1.0.0` engine.

> **Important**
>
> Memutakhirkan ke rilis mesin ini dari versi apa pun sebelumnya **`1.1.0.0`** juga memicu peningkatan sistem operasi pada semua instance di cluster DB Anda. Karena permintaan penulisan aktif yang terjadi selama pemutakhiran sistem operasi tidak akan diproses, Anda

harus menjeda semua beban kerja tulis ke klaster yang sedang ditingkatkan, termasuk pemuatan data massal, sebelum memulai pemutakhiran.
Pada awal pemutakhiran, Neptunus menghasilkan snapshot dengan nama yang terdiri dari diikuti oleh pengidentifikasi yang dibuat `preupgrade` secara otomatis berdasarkan informasi cluster DB Anda. Anda tidak akan dikenakan biaya untuk snapshot ini, dan Anda dapat menggunakannya untuk memulihkan cluster DB Anda jika terjadi kesalahan selama proses peningkatan.
Ketika upgrade mesin itu sendiri telah selesai, versi mesin baru akan tersedia sebentar pada sistem operasi lama, tetapi dalam waktu kurang dari 5 menit semua instance di cluster Anda secara bersamaan akan memulai upgrade sistem operasi. Cluster DB Anda tidak akan tersedia pada saat ini selama sekitar 6 menit. Anda dapat melanjutkan beban kerja menulis setelah peningkatan selesai.
Proses ini menghasilkan peristiwa berikut:

- Pesan acara per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Pesan acara per instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> Dimulai dengan rilis mesin ini, Neptunus tidak lagi R4 mendukung jenis instans. Jika Anda menggunakan R4 instance di cluster DB Anda, Anda harus menggantinya secara manual dengan jenis instans yang berbeda sebelum memutakhirkan ke rilis ini. Jika contoh penulis Anda adalahR4, ikuti instruksi ini untuk memindahkannya.

Peningkatan ke Rilis Ini

Amazon Neptunus 1.1.0.0.R2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.0.5.1 (2021-10-01)

Pada 2021-10-01, versi mesin 1.0.5.1 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis: 1.0.5.1.R2 (2021-10-26)
- Rilis: 1.0.5.1.R3 (2022-01-13)
- Rilis pemeliharaan: 1.0.5.1.R4 (2022-05-16)

Fitur Baru dalam Rilis Mesin Ini

- Ditambahkan hasil cache untuk caching hasil query tertentu.
- Ditambahkan Date/time dukungan di Neptune OpenCypher.
- Ditambahkan dukungan untuk `List` dan `Map` akses ke elemen di Neptunus OpenCypher.

Perbaikan dalam Rilis Mesin Ini

- Membuat nama titik akhir Neptunus OpenCypher tidak peka huruf besar/kecil.
- Peningkatan OpenCypher menjelaskan.
- Peningkatan pola kueri upsert tunggal Gremlin berakhir dengan dan langkah-langkah. `iterate()` `profile()`
- Peningkatan kinerja di Gremlin `keys()` dan `property()` fungsi.
- `dedup()`Langkah Gremlin dijalankan di DFE ketika digunakan dengan lingkup global.
- HASPredikat Gremlin berikut dijalankan di mesin DFE saat mesin DFE diaktifkan:
  - `EQ`
  - `NEQ`
  - `LT`
  - `LTE`
  - `GT`
  - `GTE`
  - `BETWEEN`
  - `INSIDE`
  - `OUTSIDE`
  - `WITHIN`
  - `AND (connectives)`

  - `OR (connectives)`
- Peningkatan kinerja kueri LIMIT.
- Peningkatan kinerja kueri agregasi umum OpenCypher.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin yang memungkinkan tepi terhubung ke tepi lain.
- Memperbaiki bug Gremlin yang menyebabkan strategi bergabung yang kurang optimal dipilih.
- Memperbaiki bug Gremlin yang menyebabkan serialisasi node dan hubungan terhenti ketika ada lebih dari 100 properti.
- Memperbaiki bug yang memperlambat perencanaan eksekusi kueri untuk kueri dengan pola grafik besar.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.0.5.1, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.11`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.0.5.1

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Anda tidak akan secara otomatis meningkatkan ke rilis ini.

Peningkatan ke Rilis Ini

Amazon Neptune 1.0.5.1 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Rilis pemeliharaan Amazon Neptunus, versi 1.0.5.1.R4 (2022-05-16)

Pada 2022-05-16, versi mesin 1.0.5.1.R4 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.0.5.1.R4, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.11`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.0.5.1.R4

Cluster Anda akan ditingkatkan ke rilis patch pemeliharaan ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi `1.0.5.1` engine.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptunus 1.0.5.1.R4 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.0.5.1.R3 (2022-01-13)

Pada 2022-01-13, versi mesin 1.0.5.1.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug yang dapat menyebabkan kebocoran sumber daya saat kueri gagal memperoleh semua sumber daya yang dibutuhkannya.
- Memperbaiki kebocoran memori kecil selama eksekusi kueri yang disebabkan oleh alokasi memori yang tidak diklaim.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.0.5.1.R3, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.11`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.0.5.1.R3

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.5.1`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptunus 1.0.5.1.R3 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.0.5.1.R2 (2021-10-26)

Pada 2021-10-26, versi mesin 1.0.5.1.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug yang menyebabkan server restart ketika kesalahan sementara terjadi saat membuat versi lama dari elemen grafik, di bawah isolasi baca berulang. Neptunus sekarang mengembalikan kesalahan, sehingga klien dapat mencoba lagi.
- Memperbaiki bug yang menyebabkan server restart ketika kesalahan sementara terjadi selama pembaruan kardinalitas tunggal. Neptunus sekarang mengembalikan kesalahan, sehingga klien dapat mencoba lagi.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.0.5.1.R2, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.11`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.0.5.1.R2

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.5.1`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptunus 1.0.5.1.R2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.0.5.0 (2021-07-27)

Pada 2021-07-27, engine versi 1.0.5.0 sedang digunakan secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis: 1.0.5.0.R2 (2021-08-16)
- Rilis: 1.0.5.0.R3 (2021-09-15)
- Rilis pemeliharaan: 1.0.5.0.R5 (2022-05-16)

Fitur Baru dalam Rilis Mesin Ini

- Neptunus ML dirilis untuk penggunaan produksi dengan banyak fitur baru, dan tidak lagi dalam mode lab.
- Menambahkan dukungan awal untuk bahasa OpenCypher query, dalam Mode Lab. OpenCypher adalah standar open-source untuk bahasa query Cypher. Sintaksnya ditentukan dalam Referensi Bahasa Query Cypher (Versi 9), dan dikelola oleh proyek OpenCypher.

  Lihat Mengakses Grafik Neptunus dengan OpenCypher untuk informasi tentang implementasi bahasa Neptunus.

  Support untuk protokol Bolt, yang digunakan klien Neptune untuk kueri OpenCypher, juga didukung. Lihat Menggunakan protokol Bolt untuk membuat kueri OpenCypher ke Neptunus.

  Support untuk OpenCypher sekarang diaktifkan secara otomatis, tetapi itu tergantung padaMesin DFE Neptune, yang saat ini hanya tersedia dalam mode lab. `DFEQueryEngine`Pengaturan default

dalam parameter cluster `neptune_lab_mode` DB sekarang`DFEQueryEngine=viaQueryHint`, yang berarti bahwa mesin diaktifkan tetapi hanya digunakan untuk kueri yang memiliki petunjuk `useDFE` kueri yang ada dan disetel ke. `true` Jika Anda menonaktifkan mesin DFE dengan pengaturan`DFEQueryEngine=disabled`, Anda tidak akan dapat menggunakan OpenCypher.

- Ditambahkan dukungan untuk SPARQL 1.1 Graph Store HTTP Protocol. Lihat Menggunakan Protokol HTTP (GSP) SPARQL 1.1 Graph Store di Amazon Neptunus.
- Mengubah pengaturan mode lab default untuk Mesin DFE Neptune to`viaQueryHint`, yang berarti bahwa mesin DFE sekarang diaktifkan secara default, tetapi hanya digunakan untuk kueri yang memiliki petunjuk kueri yang ada dan `useDFE` disetel ke. `true`
- Menambahkan CloudWatch metrik Amazon baru,`StatsNumStatementsScanned`, untuk memantau perhitungan statistik untuk mesin DFE Neptunus. Lihat Menggunakan `StatsNumStatementsScanned` CloudWatch metrik untuk memantau perhitungan statistik.

Perbaikan dalam Rilis Mesin Ini

- Ditambahkan dukungan untuk Apache TinkerPop 3.4.11.

> **Important**
>
> Perubahan dibuat di TinkerPop versi 3.4.11 yang meningkatkan kebenaran bagaimana kueri diproses, tetapi untuk saat ini terkadang dapat berdampak serius pada kinerja kueri. Misalnya, kueri semacam ini dapat berjalan jauh lebih lambat:
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
>   out()
> ```
>
> Simpul setelah langkah batas sekarang diambil dengan cara yang tidak optimal karena perubahan 3.4.11. TinkerPop Untuk menghindari hal ini, Anda dapat memodifikasi kueri dengan menambahkan langkah penghalang () kapan saja setelah`order().by()`. Misalnya:
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
> ```

```
    limit(10).
    barrier().
    out()
```

- Petunjuk `joinOrder` kueri SPARQL sekarang didukung oleh mesin kueri alternatif Neptunus DFE.
- Output dari API status Neptunus telah diperluas dan direorganisasi untuk memberikan kejelasan lebih lanjut tentang pengaturan dan fitur cluster DB Anda.

  Output baru memiliki `features` objek tingkat atas yang berisi informasi status tentang fitur cluster DB Anda, dan `settings` objek tingkat atas yang berisi informasi pengaturan. Untuk meninjau format baru, lihatContoh output dari perintah status instance.
- Penanganan log perubahan streaming telah ditingkatkan ketika `AFTER_SEQUENCE_NUMBER` aliran diminta dengan ID peristiwa terakhir di server, ketika ID peristiwa tersebut telah kedaluwarsa. Server tidak lagi membuang kesalahan ID peristiwa yang kedaluwarsa jika ID peristiwa yang diminta adalah ID peristiwa yang paling baru dibersihkan pada server.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin yang terkait dengan urutan nilai numerik.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan kluster DB ke versi 1.0.5.0, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.11`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.0.5.0

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Anda tidak akan secara otomatis meningkatkan ke rilis ini.

Peningkatan ke Rilis Ini

Amazon Neptune 1.0.5.0 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Rilis pemeliharaan Amazon Neptunus, versi 1.0.5.0.R5 (2022-05-16)

Pada 2022-05-16, versi mesin 1.0.5.0.R5 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.0.5.0.R5, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.11`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.0.5.0.R5

Cluster Anda akan ditingkatkan ke rilis patch pemeliharaan ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan engine versi 1.0.5.0.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptunus 1.0.5.0.R5 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.0.5.0.R3 (2021-09-15)

Pada 2021-09-15, versi mesin 1.0.5.0.R3 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug yang menyebabkan mesin menjadi tidak responsif dalam salah satu situasi ini:
  - Beban massal terjadi pada saat yang sama dengan perhitungan statistik otomatis sedang berlangsung.
  - Perhitungan statistik diminta secara manual pada saat yang sama saat yang sudah terjadi.
- Memperbaiki bug dalam deteksi kebuntuan dan akuisisi kunci yang dapat menyebabkan mesin mogok.
- Memperbaiki bug Gremlin di mana mesin melempar kesalahan saat menemukan data yang tidak dikenal dari titik akhir ML jarak jauh dalam kueri inferensi Gremlin.
- Memperbaiki beberapa bug dalam manajemen model ML APIs yang terkait dengan pekerjaan transformasi model dan rekomendasi instance.
- Memperbaiki bug yang dapat menyebabkan mesin mogok saat menghasilkan node dan edge IDs.
- Memperbaiki bug yang memperlambat pembuatan rencana kueri untuk kueri dengan pola grafik besar.
- Memperbaiki bug OpenCypher yang dapat menyebabkan kueri terhenti saat mengambil node yang memiliki lebih dari 100 properti.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.0.5.0.R3, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.11`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.0.5.0.R3

Cluster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan engine versi 1.0.5.0.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptune 1.0.5.0.R3 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Mesin Amazon Neptunus Versi 1.0.5.0.R2 (2021-08-16)

Pada 2021-08-16, versi mesin 1.0.5.0.R2 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan Cacat dalam Rilis Mesin Ini

- Menonaktifkan pengoptimalan yang dibuat dalam rilis mesin `1.0.5.0` yang membuat cache pencarian Neptunus bertahan dari restart mesin pada replika. Replica restart sekarang menghapus cache pencarian.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.0.5.0.R2, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.11`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.0.5.0.R2

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.5.0`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptunus 1.0.5.0.R2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.4.2 (2021-06-01)

> **Note**
>
> Mesin rilis versi 1.0.4.2.R2 adalah versi pertama dari 1.0.4.2 yang sebenarnya akan dirilis.

Topik

- Mesin Amazon Neptunus Versi 1.0.4.2.R5 (2021-08-16)
- Versi Mesin Amazon Neptune 1.0.4.2.R4 (2021-07-23)
- Versi Mesin Amazon Neptune 1.0.4.2.R3 (2021-06-28)
- Versi Mesin Amazon Neptune 1.0.4.2.R2 (2021-06-01)
- Versi Mesin Amazon Neptune 1.0.4.2.R1 (2021-05-27)

Mesin Amazon Neptunus Versi 1.0.4.2.R5 (2021-08-16)

Pada 2021-08-16, versi mesin 1.0.4.2.R5 umumnya digunakan. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan Cacat dalam Rilis Mesin Ini

- Menonaktifkan pengoptimalan yang dibuat dalam rilis mesin `1.0.4.2.R4` yang membuat cache pencarian Neptunus bertahan dari restart mesin pada replika. Replica restart sekarang menghapus cache pencarian.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum memutakhirkan cluster DB ke versi 1.0.4.2.R5, pastikan proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.10`
- Versi SPARQL: `1.1`

Tingkatkan Jalur ke Rilis Mesin 1.0.4.2.R5

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.4.2`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Versi Mesin Amazon Neptune 1.0.4.2.R4 (2021-07-23)

Per 2021-07-23, versi mesin 1.0.4.2.R4 umumnya sedang di-deploy. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan dalam Rilis Mesin Ini

- Perbaikan perilaku cache pencarian guna menghindari pengapusan cache berlebihan setelah menjalankan reset cepat pada replika.
- Perbaikan penanganan log perubahan streaming saat `AFTER_SEQUENCE_NUMBER` aliran diminta dengan ID peristiwa terakhir di server, ketika ID peristiwa itu telah kedaluwarsa. Server tidak lagi membuang kesalahan ID peristiwa yang kedaluwarsa jika ID peristiwa yang diminta adalah ID peristiwa yang paling baru dibersihkan pada server.

Perbaikan Cacat dalam Rilis Mesin Ini

- Perbaikan bug yang diperkenalkan di 1.0.4.0.R1 dimana query tidak akan mengembalikan keseluruhan nilai string lebih besar dari 760 karakter. Istilah yang terpengaruh oleh bug ini adalah literal RDF dan URIs, atau Gremlin, kunci IDs, dan nilai string.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.4.2.R4, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.10`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.4.2.R4

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.4.2`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Versi Mesin Amazon Neptune 1.0.4.2.R3 (2021-06-28)

Per 2021-06-28, versi mesin 1.0.4.2.R3 secara umum sedang di-deploy. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Masalah yang diketahui dalam rilis mesin ini

Masalah:

Sebuah bug SPARQL yang gagal menghormati tipe media dalam header `Accept` jika ada spasi.

Sebagai contoh, kueri dengan `-H "Accept: text/csv; q=1.0, */*; q=0.1"` mengembalikan output JSON bukan output CSV.

Solusi:

Jika Anda menghapus spasi di klausul `Accept` di header, mesin mengembalikan output dalam format benar yang diminta. Dengan kata lain, bukan `-H "Accept: text/csv; q=1.0, */*; q=0.1"`, gunakan:

```
-H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug dalam pembersihan cache pencarian pada replika setelah reset cepat.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.4.2.R3, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.10`
- Versi SPARQL: `1.1`

Jalur Peningkatan ke Rilis Mesin 1.0.4.2.R3

Rilis Patch ini opsional kecuali klaster DB Anda menggunakan satu atau beberapa instans R5d. Jika klaster Anda memiliki R5d instans, klaster tersebut secara otomatis akan ditingkatkan di jendela pemeliharaan berikutnya. Jika tidak, klaster itu tidak akan secara otomatis ditingkatkan ke rilis patch ini.

Anda dapat memutakhirkan 1.0.4.2.R3 rilis 1.0.4.2.R2 ke rilis ini secara manual menggunakan AWS CLI apply-pending-maintenance-actionperintah (ApplyPendingMaintenanceActionAPI).

Versi Mesin Amazon Neptune 1.0.4.2.R2 (2021-06-01)

Per 2021-06-01, versi mesin 1.0.4.2.R2 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis: 1.0.4.2.R3 (2021-06-28)

Masalah yang diketahui dalam rilis mesin ini

Masalah:

Sebuah bug SPARQL yang gagal menghormati tipe media dalam header Accept jika ada spasi.

Sebagai contoh, kueri dengan `-H "Accept: text/csv; q=1.0, */*; q=0.1"` mengembalikan output JSON bukan output CSV.

Solusi:

Jika Anda menghapus spasi di klausul Accept di header, mesin mengembalikan output dalam format benar yang diminta. Dengan kata lain, bukan `-H "Accept: text/csv; q=1.0, */*; q=0.1"` , gunakan:

```
-H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

Fitur Baru dalam Rilis Mesin Ini

- Menambahkan tipe instans R5d baru, yang mencakup cache pencarian untuk mempercepat membaca dalam penggunaan kasus melibatkan volume tinggi nilai properti atau pencarian literal RDF. Lihat Cache pencarian Neptune dapat mempercepat kueri baca.

- Menambahkan parameter mode lab baru yang memungkinkan mesin DFE eksperimental dipanggil hanya pada basis per-kueri dengan `useDFE` petunjuk kueri.

Perbaikan dalam Rilis Mesin Ini

- Menambahkan dukungan untuk TinkerPop 3.4.10.
- Penambahan dukungan untuk menggunakan langkah konfigurasi `withStrategies( )` saat mengirim permintaan tulisan Gremlin. Secara khusus, `SubgraphStrategy`, `PartitionStrategy`, `ReadOnlyStrategy`, `EdgeLabelVerificationStrategy`, dan `ReservedKeysVerificationStrategy` didukung.
- Optimalisasi tambahan untuk `V()` traversal di tengah-tengah kueri. Sebelumnya, traversal semacam itu tidak dioptimalkan di Neptune.
- Ditambahkan dukungan untuk RFC 2141 URNs untuk digunakan sebagai `baseUri` dan `namedGraphUri` parameter untuk beban massal.

Perbaikan Cacat dalam Rilis Mesin Ini

- Perbaikan bug Gremlin di parser dimana kueri yang salah diperlakukan sebagai valid.
- Perbaikan bug Gremlin saat membuka sebuah efek samping `aggregate()` dengan `cap().unfold()` ke `valueMap()` akan memunculkan sebuah pengecualian.
- Perbaikan bug Gremlin saat beberapa langkah `property()` setelah langkah `addV()` gagal dengan kesalahan “tidak dapat ditampilkan ke String”.
- Perbaikan bug Gremlin untuk mencegah beberapa pola insert bersyarat dari memunculkan pengecualian modifikasi bersamaan.
- Perbaikan bug Gremlin sehingga waktu habis permintaan kueri sekarang tidak dapat melebihi waktu habis sesi.
- Perbaikan bug SPARQL saat pembaruan menggunakan LOAD atau UNLOAD bisa gagal dengan HTTP kode 500 alih-alih HTTP kode 400 ketika server jauh tidak tersedia.
- Perbaikan bug saat panggilan API pengaliran gagal ketika nilai `commitNum` atau `opNum` lebih besar dari batas integer berkode 32-bit (2,147,483,647) digunakan.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.4.2.R2, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.10`
- Versi SPARQL: `1.1`

Jalur peningkatan untuk Rilis Mesin 1.0.4.2.R2

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Anda tidak akan secara otomatis meningkatkan ke rilis ini.

Peningkatan ke Rilis Ini

Amazon Neptune 1.0.4.2.R2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.2 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.2 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.4.2.R1 (2021-05-27)

Rilis mesin 1.0.4.2.R1 tidak pernah di-deploy.

Mesin Amazon Neptune Versi 1.0.4.1 (2020-12-08)

Per 2020-12-08, versi mesin 1.0.4.1 umumnya di-deploy. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis: 1.0.4.1.R1.1 (2021-03-22)
- Rilis: 1.0.4.1.R2 (2021-02-24)

Important

Rilis: 1.0.4.0 (2020-10-12) membuat TLS 1.2 dan HTTPS wajib untuk semua koneksi ke Amazon Neptune. Namun, bug dalam rilis itu telah memungkinkan koneksi HTTP koneksi TLS and/or usang untuk terus bekerja bagi pelanggan yang sebelumnya menetapkan parameter cluster DB untuk mencegah penegakan koneksi HTTPS.
Bug itu diperbaiki dalam rilis patch 1.0.4.0.R2 dan 1.0.4.1.R2, tetapi perbaikan tersebut telah menyebabkan kegagalan koneksi tak terduga ketika patch diinstal secara otomatis. Untuk alasan ini, kedua patch telah dikembalikan, dan hanya dapat diinstal secara manual, untuk memberi Anda kesempatan memperbarui pengaturan Anda untuk TLS 1.2.
Harus digunakan SSL/TLS untuk semua koneksi ke Neptunus memengaruhi koneksi Anda dengan konsol Gremlin, driver Gremlin, Python Gremlin, .NET, NodeJS, REST, dan juga koneksi penyeimbang beban. APIs Jika Anda telah menggunakan HTTP atau versi TLS yang lebih lama untuk salah satu atau semua sampai sekarang, Anda harus memperbarui

klien dan driver yang relevan dan mengubah kode Anda untuk menggunakan HTTPS secara eksklusif sebelum memperbarui sistem Anda ke patch terbaru.

Fitur Baru dalam Rilis Mesin Ini

- Memperkenalkan fitur Neptune ML, yang menghadirkan kemampuan machine learning yang ampuh ke Amazon Neptune. Lihat Amazon Neptunus ML untuk pembelajaran mesin pada grafik.
- Menambahkan operasi `UNLOAD` SPARQL kustom untuk menghapus data yang diambil dari sumber remote. Lihat SPARQL UPDATE UNLOAD.

Perbaikan dalam Rilis Mesin Ini

- Pengoptimalan beberapa Gremlin pola insert bersyarat untuk menghindari pengecualian concurrent-modification.

Perbaikan Cacat dalam Rilis Mesin Ini

- Perbaikan bug Gremlin yang dapat menyebabkan hasil hilang untuk pola spesifik dari query yang menggunakan langkah `as()`.
- Perbaikan bug Gremlin yang dapat menyebabkan kesalahan ketika menggunakan langkah `project()` yang bersarang di dalam langkah lain seperti `union()`.
- Perbaikan bug Gremlin di langkah `project()`.
- Perbaikan bug Gremlin di dalam traversal berbasis string dimana langkah `none()` tidak bekerja.
- Perbaikan bug Gremlin dalam traversal berbasis string dimana sebuah peta kosong tidak didukung sebagai argumen ke langkah `inject()`.
- Perbaikan bug Gremlin dalam eksekusi traversal berbasis string di mesin DFE dimana metode terminal seperti `toList()` tidak bekerja dengan baik.
- Perbaikan bug Gremlin yang gagal menutup transaksi yang menggunakan langkah `iterate()` dalam query String.
- Perbaikan bug Gremlin yang dapat menyebabkan kueri menggunakan pola `is(P.gte(0))` untuk membuang pengecualian dalam beberapa situasi.
- Perbaikan bug Gremlin yang dapat menyebabkan kueri menggunakan pola `order().by(T.id)` untuk membuang pengecualian dalam beberapa situasi.

- Perbaikan bug Gremlin yang dapat menyebabkan kueri menggunakan pola `addV().aggregate()` untuk memberi hasil yang salah dalam beberapa situasi.
- Perbaikan bug Gremlin yang dapat menyebabkan kueri menggunakan langkah `path()` diikuti oleh pola langkah `project()` untuk membuang pengecualian dalam beberapa situasi.
- Perbaikan bug SPARQL saat fungsi `SUBSTR` menandakan kesalahan alih-alih mengembalikan string kosong.
- Perbaikan bug di mesin DFE yang dapat menyebabkan operasi gabungan dalam kueri permintaan non-blocking menghasilkan hasil yang salah saat adanya variabel tak terbatas.

Versi Query Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.4.1, pastikan bahwa proyek Anda kompatibel dengan versi query bahasa ini:

- Versi Gremlin: `3.4.8`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.4.1

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.4.1`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptune 1.0.4.1 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
```

```
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu

akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.4.1.R1.1 (2021-03-22)

Seperti 2021-03-22, versi mesin 1.0.4.1.R1.1 umumnya di-deploy. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan Cacat dalam Rilis Mesin Ini

- Menonaktifkan optimasi untuk Gremlin pola insert bersyarat yang dapat ditambahkan atau dilampirkan ke label dan properti yang ada.

Versi Query Bahasa yang Didukung dalam Rilis Ini

Sebelum meng-upgrade cluster DB ke versi 1.0.4.1.R1.1, pastikan bahwa proyek Anda kompatibel dengan versi query bahasa ini:

- Versi Gremlin: `3.4.8`

- Versi SPARQL: `1.1`

Jalur Peningkatan ke Rilis Mesin 1.0.4.1.R1.1

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi mesin`1.0.4.1`.

Meningkatkan ke Rilis Ini

Amazon Neptune 1.0.4.1.R1.1 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada jalur peningkatan untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.4.1.R2 (2021-02-24)

Seperti 2021-02-24, versi mesin 1.0.4.1.R2 umumnya di-deploy. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis: 1.0.4.1.R2.1 (2021-03-11)

Fitur Baru dalam Rilis Mesin Ini

- Neptune sekarang mendukung kompresi dari file tunggal di format `bzip2` untuk beban massal. Lihat Muat Format Data.

Perbaikan Cacat dalam Rilis Mesin Ini

- Perbaikan sebuah bug di Rilis: 1.0.4.0 (2020-10-12) yang mengizinkan koneksi ke Neptune menggunakan versi `HTTP` atau versi TLS sebelumnya, bukan`HTTPS` dan TLS 1.2.

  > **Important**
  >
  > Harus digunakan SSL/TLS untuk semua koneksi ke Neptunus bisa menjadi perubahan besar. Ini memengaruhi koneksi Anda dengan konsol Gremlin, driver Gremlin, Python Gremlin, .NET, NodeJS, REST, dan juga koneksi load-balancer. APIs Jika Anda telah menggunakan HTTP atau versi TLS yang lebih lama untuk salah satu atau semua sampai sekarang, Anda harus memperbarui klien dan driver yang relevan sebelum menginstal patch ini dan ubah kode Anda untuk memakai HTTPS secara eksklusif.

- Memperbaiki bug Gremlin di mana `InternalFailureException` ditetapkan sebagai kode respons dalam keadaan tertentu ketika `ConcurrentModificationException` terjadi.
- Memperbaiki bug Gremlin di mana dalam kondisi tertentu memperbarui edge atau vertex dapat menyebabkan `InternalFailureException` transien.

Versi Query Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.4.1.R2, pastikan bahwa proyek Anda kompatibel dengan versi query bahasa ini:

- Versi Gremlin: `3.4.8`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.4.1.R2

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menjalankan versi mesin`1.0.4.1`.

Meningkatkan ke Rilis Ini

Amazon Neptune 1.0.4.1.R2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada jalur peningkatan untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.4.1.R2.1 (2021-03-11)

Seperti 2021-03-11, versi mesin 1.0.4.1.R2.1 umumnya di-deploy. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan Cacat dalam Rilis Mesin Ini

- Menonaktifkan optimasi untuk Gremlin pola insert bersyarat yang dapat ditambahkan atau dilampirkan ke label dan properti yang ada.

Versi Query Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.4.1.R2.1, pastikan bahwa proyek Anda kompatibel dengan versi query bahasa ini:

- Versi Gremlin: `3.4.8`
- Versi SPARQL: `1.1`

Jalur Peningkatan ke Rilis Mesin 1.0.4.1.R2.1

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.4.1.R2`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptune 1.0.4.1.R2.1 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada jalur peningkatan untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi

syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1.R2 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1.R2 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.4.0 (2020-10-12)

Per 2020-10-12, versi mesin 1.0.4.0 umumnya di-deploy. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis: 1.0.4.0.R2 (2021-02-24)

Fitur Baru dalam Rilis Mesin Ini

- Tambahan kompresi frame-level untuk Gremlin.

Perbaikan dalam Rilis Mesin Ini

- Amazon Neptunus sekarang memerlukan penggunaan Secure Sockets Layer (SSL) dengan protokol .2 untuk semua koneksi ke Neptunus TLSv1 di semua wilayah, menggunakan cipher suite yang kuat ini:
  - `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
  - `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
  - `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
  - `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
  - `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
  - `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

  Ini berlaku untuk REST dan WebSocket koneksi ke Neptunus, dan berarti Anda harus menggunakan HTTPS daripada HTTP saat menghubungkan ke Neptunus di semua wilayah.

  Karena koneksi klien yang menggunakan HTTP atau TLS 1.1 tidak lagi didukung di mana saja, pastikan bahwa klien dan kode Anda telah diperbarui untuk menggunakan TLS 1.2 dan HTTPS sebelum meningkatkan ke rilis mesin ini.

> **Important**
>
> Harus digunakan SSL/TLS untuk semua koneksi ke Neptunus bisa menjadi perubahan besar. Ini memengaruhi koneksi Anda dengan konsol Gremlin, driver Gremlin, Python Gremlin, .NET, NodeJS, REST, dan juga koneksi load-balancer. APIs Jika Anda telah menggunakan HTTP untuk salah satu atau semua ini, Anda sekarang harus memperbarui klien dan driver yang relevan dan mengubah kode Anda untuk menggunakan HTTPS atau koneksi Anda akan gagal.
> Bug dalam rilis ini telah memungkinkan koneksi HTTP koneksi TLS and/or usang untuk terus bekerja bagi pelanggan yang sebelumnya menetapkan parameter cluster DB untuk mencegah penegakan koneksi HTTPS. Bug itu diperbaiki dalam rilis patch 1.0.4.0.R2 dan

1.0.4.1.R2, tetapi perbaikan tersebut telah menyebabkan kegagalan koneksi tak terduga ketika patch diinstal secara otomatis.
Untuk alasan ini, kedua patch telah dikembalikan, dan hanya dapat diinstal secara manual, untuk memberi Anda kesempatan memperbarui pengaturan Anda untuk TLS 1.2.

- Upgrade TinkerPop ke versi 3.4.8. Ini adalah upgrade kompatibel mundur. Lihat log TinkerPop perubahan untuk apa yang baru.
- Peningkatan kinerja untuk langkah `properties()` Gremlin.
- Menambahkan detail tentang `BindOp` dan `MultiplexerOp` dalam laporan explain dan profile.
- Menambahkan data prefetch untuk meningkatkan kinerja ketika ada cache terlewat.
- Menambahkan pengaturan `allowEmptyStrings` baru di parameter `parserConfiguration` loader yang memungkinkan string kosong diperlakukan sebagai nilai properti yang valid dalam beban CSV (lihat Parameter Permintaan Loader Neptune).
- Loader sekarang memungkinkan titik koma escaped dalam kolom CSV multivalue.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki potensi kebocoran memori Gremlin yang terkait dengan langkah `both()`.
- Memperbaiki bug di mana metrik permintaan hilang karena titik akhir yang berakhir di '/' tidak ditangani dengan benar.
- Memperbaiki bug yang menyebabkan replika jatuh di belakang dan restart di bawah beban berat ketika mesin DFE diaktifkan dalam mode lab.
- Memperbaiki bug yang mencegah pesan kesalahan yang benar dilaporkan saat beban massal gagal karena suatu out-of-memory kondisi.
- Memperbaiki bug SPARQL di mana pengkodean karakter ditempatkan di header Content-Encoding dalam respons kueri SPARQL. Sekarang `charset` justru ditempatkan di header Content-Type, memungkinkan klien HTTP untuk mengenali set karakter yang digunakan secara otomatis.

Versi Bahasa Kueri yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.4.0, pastikan bahwa proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.8`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.4.0

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Anda tidak akan secara otomatis meningkatkan ke rilis ini.

Peningkatan ke Rilis Ini

Amazon Neptune versi 1.0.4.0 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada jalur peningkatan untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui Dukungan AWS Premium.

Versi Mesin Amazon Neptune 1.0.4.0.R2 (2021-02-24)

Per 2021-02-24, versi mesin 1.0.4.0.R2 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan Cacat dalam Rilis Mesin Ini

- Perbaikan sebuah bug di Rilis: 1.0.4.0 (2020-10-12) yang mengizinkan koneksi ke Neptune menggunakan versi HTTP atau versi TLS sebelumnya, bukanHTTPS dan TLS 1.2.

  Important

  Harus digunakan SSL/TLS untuk semua koneksi ke Neptunus bisa menjadi perubahan besar. Ini memengaruhi koneksi Anda dengan konsol Gremlin, driver Gremlin, Python Gremlin, .NET, NodeJS, REST, dan juga koneksi load-balancer. APIs Jika Anda telah menggunakan HTTP atau versi TLS yang lebih lama untuk salah satu atau semua sampai sekarang, Anda harus memperbarui klien dan driver yang relevan sebelum menginstal patch ini dan ubah kode Anda untuk memakai HTTPS secara eksklusif.

- Memperbaiki bug dalam pemuatan massal CSV yang melibatkan label yang berakhir di #.
- Memperbaiki bug Gremlin di mana `InternalFailureException` ditetapkan sebagai kode respons dalam keadaan tertentu ketika `ConcurrentModificationException` terjadi.
- Memperbaiki bug Gremlin di mana dalam kondisi tertentu memperbarui edge atau vertex dapat menyebabkan `InternalFailureException` transien.

Versi Query Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.4.0.R2, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.8`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.4.0.R2

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.4.0`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptune 1.0.4.0.R2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada jalur peningkatan untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.3.0 (2020-08-03)

Per 2020-08-03, versi mesin 1.0.3.0 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis: 1.0.3.0.R2 (2020-10-12)
- Rilis: 1.0.3.0.R3 (2021-02-19)

Fitur Baru dalam Rilis Mesin Ini

- Neptune telah memperkenalkan mesin kueri alternatif baru (DFE) yang secara signifikan dapat mempercepat eksekusi kueri. Lihat Mesin kueri alternatif Amazon Neptune (DFE).
- DFE bergantung pada statistik yang dihasilkan sebelumnya tentang data grafik Neptune Anda yang dikelola melalui titik akhir statistik baru. Lihat Statistik DFE.
- Sekarang Anda dapat mengecualikan pekerjaan pemuatan antrian dari daftar beban yang IDs dikembalikan oleh Loader Get-Status API dengan menyetel `includeQueuedLoads` parameter baru ke FALSE. Lihat Parameter permintaan Neptune Loader Get-Status.
- Neptune sekarang mendukung header trailing untuk respons kueri SPARQL yang dapat berisi kode dan pesan kesalahan jika permintaan gagal setelah mulai mengembalikan potongan respons. Lihat Header di belakang HTTP opsional untuk respons SPARQL multi-bagian.
- Neptune sekarang juga memungkinkan Anda mengaktifkan encoding respons potongan untuk kueri Gremlin. Seperti dalam kasus SPARQL, potongan respons memiliki header trailing yang dapat berisi kode dan pesan kesalahan jika terjadi kegagalan setelah kueri telah mulai mengembalikan potongan respons. Lihat Gunakan header jejak HTTP opsional untuk mengaktifkan respons multi-bagian Gremlin.

Perbaikan dalam Rilis Mesin Ini

- Anda sekarang dapat memberikan ukuran permintaan batch ElasticSearch untuk pencarian teks lengkap di Gremlin.
- Penggunaan memori yang lebih baik untuk kueri SPARQL GROUP BY.
- Menambahkan optimizer kueri Gremlin baru untuk memangkas filter tak terbatas tertentu.
- Peningkatan waktu maksimum WebSocket koneksi yang diautentikasi menggunakan IAM dapat tetap terbuka, dari 36 jam menjadi 10 hari.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug di mana jika Anda mengirim parameter URL yang tidak dikodekan dalam permintaan POST, Neptune mengembalikan kode status HTTP 500 dan `InternalServerErrorException`. Sekarang Neptune mengembalikan kode status HTTP 400 dan `BadRequestException`, dengan pesan: `Failure to process the POST request parameters`.
- Memperbaiki bug Gremlin di mana kegagalan WebSocket koneksi tidak dilaporkan dengan benar.
- Mempebaiki bug Gremlin yang melibatkan sideEffects yang menghilang.
- Memperbaiki bug Gremlin di mana parameter `batchsize` pencarian teks-lengkap tidak didukung dengan benar.
- Memperbaiki bug Gremlin untuk menangani `toV` dan `fromV` secara individu untuk setiap arah di `bothE`.
- Perbaikan bug Gremlin yang melibatkan `Edge pathType` di langkah `hasLabel`.
- Memperbaiki bug SPARQL di mana pengurutan ulang gabungan dengan binding statis tidak bekerja dengan benar.
- Memperbaiki bug SPARQL UPDATE LOAD di mana bucket Amazon S3 yang tidak tersedia tidak dilaporkan dengan benar.
- Memperbaiki bug SPARQL di mana masalah dengan node SERVICE di subkueri tidak dilaporkan dengan benar.
- Memperbaiki bug SPARQL di mana kueri berisi syarat FILTER EXISTS atau FILTER NOT EXISTS nested yang tidak dievaluasi dengan benar.
- Memperbaiki bug SPARQL untuk menangani binding yang dihasilkan duplikat dengan benar ketika memanggil titik akhir Layanan SPARQL melalui pembuatan kueri.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.3.0, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.3`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.3.0

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Jika klaster Anda memiliki parameter `AutoMinorVersionUpgrade` yang diatur ke `True`, klaster Anda akan ditingkatkan ke rilis mesin ini secara otomatis dua sampai tiga minggu setelah tanggal rilis ini, selama jendela pemeliharaan.

Peningkatan ke Rilis Ini

Amazon Neptune 1.0.3.0 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.3.0.R3 (2021-02-19)

Per 2021-02-19, versi mesin 1.0.3.0.R3 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug dalam pemuatan massal CSV yang melibatkan label yang berakhir di #.
- Perbaikan bug Gremlin yang dapat menyebabkan hasil hilang untuk pola spesifik dari kueri yang menggunakan langkah `as()`.
- Perbaikan bug Gremlin yang dapat menyebabkan kesalahan ketika menggunakan langkah `project()` yang bersarang di dalam langkah lain seperti `union()`.
- Memperbaiki bug Gremlin dalam eksekusi traversal string di mesin DFE eksperimental ketika sebuah metode terminal seperti `toList()` digunakan.
- Perbaikan bug Gremlin yang gagal menutup transaksi saat menggunakan langkah `iterate()` dalam kueri string.
- Perbaikan bug Gremlin yang dapat menyebabkan kueri menggunakan pola `is(P.gte(0))` untuk melemparkan pengecualian dalam beberapa situasi.
- Memperbaiki bug Gremlin di mana `InternalFailureException` ditetapkan sebagai kode respons dalam keadaan tertentu ketika `ConcurrentModificationException` terjadi.
- Memperbaiki bug Gremlin di mana dalam kondisi tertentu memperbarui edge atau vertex dapat menyebabkan `InternalFailureException` transien.

Versi Query Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.3.0.R3, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.8`
- Versi SPARQL: `1.1`

Jalur Peningkatan ke Rilis Mesin 1.0.3.0.R3

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.3.0`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptune 1.0.3.0.R3 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.3.0.R2 (2020-10-12)

Per 2020-10-12, versi mesin 1.0.3.0.R2 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan dalam Rilis Mesin Ini

- Peningkatan kinerja untuk langkah `properties()` Gremlin.
- Menambahkan detail tentang `BindOp` dan `MultiplexerOp` dalam laporan explain dan profile.
- Untuk respos kueri SPARQL, menambahkan `charset` ke header Content-Type, memungkinkan klien HTTP untuk mengenali charset yang digunakan secara otomatis.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug SPARQL di mana `CancellationException` tidak ditangani.
- Memperbaiki bug SPARQL di mana kueri yang berisi optionals nested tidak bekerja dengan benar.
- Memperbaiki bug SPARQL di LOAD di mana `ConcurrentModificationException` dapat menyebabkan kueri hang.
- Memperbaiki bug SPARQL yang mencegah respons kueri agar tidak terkompresi gzip.
- Perbaikan bug Gremlin di langkah `groupBy()`.
- Memperbaiki bug Gremlin terkait dengan penggunaan langkah `aggregate()` di dalam langkah `local()`.
- Memperbaiki bug Gremlin terkait dengan menggunakan `bothE()` yang diikuti dengan predikat yang menggunakan nilai agregat.
- Memperbaiki bug Gremlin terkait dengan penggunaan langkah `bothE()` dengan langkah `repeat()`.
- Memperbaiki potensi kebocoran memori Gremlin yang terkait dengan langkah `both()`.

- Memperbaiki bug di mana metrik permintaan hilang karena titik akhir yang berakhir di '/' tidak ditangani dengan benar.
- Memperbaiki bug yang dapat meningkatkan `ThrottlingException` bahkan ketika antrean permintaan tidak penuh.
- Memperbaiki bug dalam mengambil status pemuatan ketika sebuah pemuatan gagal karena suatu alasan seperti `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE`.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.3.0.R2, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.3`
- Versi SPARQL: `1.1`

Jalur Peningkatan ke Rilis Mesin 1.0.3.0.R2

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Jika klaster Anda memiliki parameter `AutoMinorVersionUpgrade` yang diatur ke `True`, klaster Anda akan ditingkatkan ke rilis mesin ini secara otomatis dua sampai tiga minggu setelah tanggal rilis ini, selama jendela pemeliharaan.

Peningkatan ke Rilis Ini

Amazon Neptune 1.0.3.0.R2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.2.2 (2020-03-09)

Per 2020-03-09, versi mesin 1.0.2.2 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis: 1.0.2.2.R2 (2020-04-02)
- Rilis: 1.0.2.2.R3 (2020-07-22)
- Rilis: 1.0.2.2.R4 (2020-07-23)
- Rilis: 1.0.2.2.R5 (2020-10-12)
- Rilis: 1.0.2.2.R6 (2021-02-19)

Perbaikan dalam Rilis Mesin Ini

- Menambahkan informasi ke API status tentang transaksi yang sedang di-rollback. Lihat Status instans.

- Upgrade versi Apache TinkerPop ke 3.4.3.

  Versi 3.4.3 kompatibel dengan versi sebelumnya yang didukung oleh Neptune (3.4.1). Ini memperkenalkan satu perubahan kecil dalam perilaku: Gremlin tidak lagi mengembalikan kesalahan ketika Anda mencoba untuk menutup sesi yang tidak ada (lihat Mencegah kesalahan saat menutup sesi yang tidak ada).
- Menghapus kemacetan performa dalam eksekusi langkah pencarian teks lengkap Gremlin.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug SPARQL dalam penanganan pola grafik kosong dalam kueri.
- Memperbaiki bug SPARQL dalam penanganan titik koma yang tidak dikodekan dalam kueri yang dikodekan URL.
- Memperbaiki bug Gremlin dalam penanganan vertex berulang di langkah `Union`.
- Memperbaiki bug Gremlin yang menyebabkan beberapa kueri dengan `.simplePath()` atau `.cyclicPath()` di dalam `.repeat()` untuk mengembalikan hasil yang salah.
- Memperbaiki bug Gremlin yang menyebabkan `.project()` mengembalikan hasil yang salah jika traversal turunannya tidak mengembalikan solusi.
- Memperbaiki bug Gremlin di mana kesalahan dari konflik baca-tulis memunculkan `InternalFailureException` ketimbang `ConcurrentModificationException`.
- Memperbaiki bug Gremlin yang menyebabkan kegagalan `.group().by(...).by(values("property"))`.
- Memperbaiki bug Gremlin dalam output profil untuk full-text-search langkah-langkah.
- Memperbaiki kebocoran sumber daya dalam sesi Gremlin.
- Memperbaiki bug yang mencegah API status melaporkan versi tertib yang benar dalam beberapa kasus.
- Memperbaiki bug loader massal yang mengizinkan URL ke lokasi selain Amazon S3 untuk digunakan sebagai sumber dalam permintaan muatan massal.
- Memperbaiki bug loader massal dalam status pemuatan terperinci.

Versi Bahasa Kueri yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.2.2, pastikan bahwa proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.3`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.2.2

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Jika klaster Anda memiliki parameter `AutoMinorVersionUpgrade` yang diatur ke `True`, klaster Anda akan ditingkatkan ke rilis mesin ini secara otomatis dua sampai tiga minggu setelah tanggal rilis ini, selama jendela pemeliharaan.

Peningkatan ke Rilis Ini

Amazon Neptune 1.0.2.2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.2.2.R6 (2021-02-19)

Per 2021-02-19, versi mesin 1.0.2.2.R6 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin di mana `InternalFailureException` ditetapkan sebagai kode respons dalam keadaan tertentu ketika `ConcurrentModificationException` terjadi.
- Memperbaiki bug Gremlin di mana dalam kondisi tertentu memperbarui edge atau vertex dapat menyebabkan `InternalFailureException` transien.

Versi Query Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.2.2.R6, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.8`
- Versi SPARQL: `1.1`

Jalur Peningkatan ke Rilis Mesin 1.0.2.2.R6

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.2.2`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptune 1.0.2.2.R6 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.2.2.R5 (2020-10-12)

Per 2020-10-12, versi mesin 1.0.2.2.R5 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan dalam Rilis Mesin Ini

- Peningkatan kinerja untuk langkah `properties()` Gremlin.
- Menambahkan detail tentang `BindOp` dan `MultiplexerOp` dalam laporan explain dan profile.
- Untuk respos kueri SPARQL, menambahkan `charset` ke header Content-Type, memungkinkan klien HTTP untuk mengenali charset yang digunakan secara otomatis.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug SPARQL di mana `CancellationException` tidak ditangani.
- Memperbaiki bug SPARQL di mana kueri yang berisi optionals nested tidak bekerja dengan benar.
- Memperbaiki bug SPARQL di LOAD di mana `ConcurrentModificationException` dapat menyebabkan kueri hang.
- Memperbaiki bug SPARQL yang mencegah respons kueri agar tidak terkompresi gzip.
- Perbaikan bug Gremlin di langkah `groupBy()`.
- Memperbaiki bug Gremlin terkait dengan penggunaan langkah `aggregate()` di dalam langkah `local()`.
- Memperbaiki bug Gremlin terkait dengan menggunakan `bothE()` yang diikuti dengan predikat yang menggunakan nilai agregat.
- Memperbaiki bug Gremlin terkait dengan penggunaan langkah `bothE()` dengan langkah `repeat()`.
- Memperbaiki potensi kebocoran memori Gremlin yang terkait dengan langkah `both()`.
- Memperbaiki bug di mana metrik permintaan hilang karena titik akhir yang berakhir di '/' tidak ditangani dengan benar.
- Memperbaiki bug yang dapat meningkatkan `ThrottlingException` bahkan ketika antrean permintaan tidak penuh.
- Memperbaiki bug dalam mengambil status pemuatan ketika sebuah pemuatan gagal karena suatu alasan seperti `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE`.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.2.2.R5, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.3`
- Versi SPARQL: `1.1`

Jalur Peningkatan ke Rilis Mesin 1.0.2.2.R5

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.2.2`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptune 1.0.2.2.R5 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.2.2.R4 (2020-07-23)

Per 2020-07-23, versi mesin 1.0.2.2.R4 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan dalam Rilis Mesin Ini

- Peningkatan penggunaan memori dengan melepaskan memori yang tidak terpakai kembali ke sistem operasi lebih sering.
- Juga penggunaan memori yang lebih baik untuk kueri SPARQL GROUP BY.
- Peningkatan waktu maksimum WebSocket koneksi dapat tetap terbuka yang diautentikasi menggunakan IAM, dari 36 jam menjadi 10 hari.
- Menambahkan `BufferCacheHitRatio` CloudWatch metrik, yang dapat berguna dalam mendiagnosis latensi kueri dan menyetel jenis instance. Lihat Metrik Neptunus.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug dalam menutup koneksi WebSocket IAM yang menganggur atau kedaluwarsa. Neptune sekarang mengirim bingkai tertutup sebelum menutup koneksi.
- Memperbaiki bug SPARQL dalam evaluasi kueri yang berisi kondisi FILTER EXISTS NEXISTS NOT EXISTS NEXISTS and/or .
- Memperbaiki bug pengakhiran kueri SPARQL yang menyebabkan utas diblokir di server dalam kondisi ekstrim tertentu.
- Perbaikan bug Gremlin yang melibatkan Edge pathType di langkah `hasLabel`.
- Memperbaiki bug Gremlin untuk menangani `toV` dan `fromV` secara individu untuk setiap arah di `bothE`.

- Mempebaiki bug Gremlin yang melibatkan sideEffects yang menghilang.

Versi Bahasa Kueri yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.2.2.R4, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.3`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.2.2.R4

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.2.2`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptune 1.0.2.2.R4 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
```

```
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.2.2.R3 (2020-07-22)

Rilis mesin 1.0.2.2.R3 dimasukkan ke dalam rilis mesin 1.0.2.2.R4.

Versi Mesin Amazon Neptune 1.0.2.2.R2 (2020-04-02)

Per 2020-04-02, versi mesin 1.0.2.2.R2 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan dalam Rilis Mesin Ini

- Sekarang Anda dapat mengantrekan hingga 64 tugas pemuatan massal, alih-alih harus menunggu satu tugas selesai sebelum memulai tugas berikutnya. Anda juga dapat melakukan eksekusi permintaan pemuatan mengantre bergantung pada keberhasilan penyelesaian satu atau beberapa tugas beban yang sebelumnya mengantre menggunakan parameter `dependencies` perintah `load`. Lihat Perintah Loader Neptune.
- Full-text-search output sekarang dapat diurutkan (lihatParameter pencarian teks lengkap).
- Sekarang ada parameter klaster DB untuk meminta aliran Neptune, dan fitur tersebut telah dipindahkan dari Mode Lab. Lihat Mengaktifkan Neptune Streams.

Perbaikan Cacat dalam Rilis Mesin Ini

- Perbaikan kegagalan stochastic di server startup menunda pembuatan instans.

- Memperbaiki masalah pengoptimasi di mana pernyataan `BIND` dalam kueri membuat optimizer memulai dengan pola tidak selektif dalam perencanaan gabungan-urutan.

Versi Bahasa Kueri yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.2.2.R2, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.3`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.2.2.R2

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.2.2`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Amazon Neptune 1.0.2.2.R2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
```

```
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.2.1 (2019-11-22)

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis: 1.0.2.1.R6 (2020-04-22)
- Rilis: 1.0.2.1.R5 (2020-04-22) Rilis patch ini tidak di-deploy.
- Rilis: 1.0.2.1.R4 (2019-12-20)
- Rilis: 1.0.2.1.R3 (2019-12-12)
- Rilis: 1.0.2.1.R2 (2019-11-25)

Fitur Baru Dalam Rilis Mesin Ini

- Menambahkan kemampuan pencarian teks lengkap melalui integrasi dengan OpenSearch Layanan Amazon. Lihat Pencarian teks lengkap Neptunus
- Menambahkan opsi menggunakan mode lab untuk membuat indeks keempat (indeks OSGP) untuk sejumlah besar predikat. Lihat Indeks OSGP.
- Menambahkan mode detail ke Explain SPARQL. Untuk detailnya, lihat Menggunakan `explain` SPARQL dan Output mode rincian.
- Menambahkan informasi mode lab ke laporan status engine. Lihat Status instans untuk detail.

- Snapshot DB Cluster sekarang dapat disalin di seluruh AWS Wilayah. Lihat Menyalin Snapshot.

Perbaikan dalam Rilis Mesin Ini

- Peningkatan performa saat menangani sejumlah besar predikat.
- Peningkatan optimalisasi kueri. Meskipun ini harus sepenuhnya transparan bagi pelanggan, kami mendorong Anda untuk menguji aplikasi Anda sebelum melakukan peningkatan untuk memastikan bahwa aplikasi berperilaku seperti yang diharapkan.
- Penyempurnaan kecil untuk pelaporan kesalahan.
- Menambahkan optimalisasi untuk langkah-langkah `.project()` dan `.identity()` Gremlin.
- Menambahkan optimalisasi untuk kasus `.union()` non-terminal Gremlin.
- Menambahkan dukungan asli untuk traversal `.path().by()` Gremlin.
- Menambahkan dukungan asli untuk `.coalesce()` Gremlin.
- Optimalisasi lebih lanjut dari penulisan massal.
- Kami sekarang mengharuskan koneksi HTTPS menggunakan setidaknya TLS versi 1.2 atau lebih tinggi, untuk mencegah outdated/insecure cipher digunakan.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug penanganan traversal di dalam `addE()` Gremlin.
- Memperbaiki bug Gremlin yang disebabkan oleh anotasi AST bocor dari traversals turunan ke induk.
- Memperbaiki bug yang terjadi di Gremlin saat `.otherV()` dipanggil setelah `select()`.
- Memperbaiki bug Gremlin yang menyebabkan beberapa langkah `.hasLabel()` gagal jika bug muncul setelah langkah `bothE()`.
- Membuat perbaikan kecil untuk .sum() dan .project() Gremlin.
- Memperbaiki bug dalam memproses kueri SPARQL yang tidak memiliki penjepit penutup.
- Memperbaiki beberapa bug kecil di Explain SPARQL.
- Memperbaiki bug dalam penanganan permintaan status pemuatan bersamaan.
- Mengurangi memori yang digunakan untuk mengeksekusi beberapa traversal Gremlin dengan langkah `.project()`.
- Memperbaiki perbandingan numerik dari nilai-nilai khusus di SPARQL. Lihat Kepatuhan Standar.

Versi Bahasa Kueri yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.2.1, pastikan bahwa proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.1`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.2.1

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Anda tidak akan secara otomatis meningkatkan ke rilis ini.

Peningkatan ke Rilis Ini

Amazon Neptune 1.0.2.1 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.2.1.R6 (2020-04-22)

Per 2020-04-22, versi mesin 1.0.2.1.R6 sedang di-deploy secara umum. Harap dicatat bahwa perlu beberapa hari agar rilis baru tersedia di setiap wilayah.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki sebuah bug di mana `ConcurrentModificationConflictException` dan `TransactionException` tidak dikonversi menjadi `NeptuneGremlinException`, menyebabkan `InternalFailureException` dikembalikan ke pelanggan.
- Memperbaiki bug di mana Neptune melaporkan statusnya sebagai sehat sebelum server benar-benar siap.
- Perbaikan bug di mana komit kamus dan transaksi pengguna rusak ketika dua pemetaan `value->id` sedang dimasukkan secara bersamaan.
- Perbaikan bug dalam serialisasi status pemuatan.
- Memperbaiki bug sesi Gremlin.
- Memperbaiki bug di mana Neptune gagal melemparkan pengecualian ketika server gagal untuk memulai.
- Memperbaiki bug di mana Neptune gagal mengirim bingkai tutup soket Web sebelum menutup saluran.

Versi Bahasa Kueri yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.2.1.R6, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.1`

- Versi SPARQL: `1.1`

Jalur Peningkatan ke Rilis Mesin 1.0.2.1.R6

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.2.1`.

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Meningkatkan ke Rilis Ini

Mesin Amazon Neptune versi 1.0.2.1.R6 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.2.1.R5 (2020-04-22)

Rilis mesin 1.0.2.1.R5 tidak pernah di-deploy.

Versi Mesin Amazon Neptune 1.0.2.1.R4 (2019-12-20)

Perbaikan dalam Rilis Mesin Ini

- Neptunus sekarang mencoba untuk selalu menempatkan panggilan full-text-search apa pun terlebih dahulu di pipeline eksekusi. Ini mengurangi volume panggilan ke OpenSearch, yang secara signifikan dapat meningkatkan kinerja. Lihat Full-text-search eksekusi kueri.
- Neptune sekarang memunculkan `IllegalArgumentException` jika Anda mencoba mengakses properti, vertex, atau edge yang tidak ada. Sebelumnya, Neptune memunculkan `UnsupportedOperationException` dalam situasi itu.

  Sebagai contoh, jika Anda mencoba menambahkan sebuah edge yang mereferensikan sebuah vertex yang tidak ada, Anda sekarang akan memunculkan `IllegalArgumentException`.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin di mana traversal `union` di dalam `project-by` tidak mengembalikan hasil atau mengembalikan hasil yang salah.
- Memperbaiki bug Gremlin yang menyebabkan langkah `.project().by()` nested untuk mengembalikan hasil yang salah.

Versi Bahasa Kueri yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.2.1.R4, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.1`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.2.1.R4

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Namun, pembaruan otomatis ke rilis ini tidak didukung.

Peningkatan ke Rilis Ini

Mesin Amazon Neptune versi 1.0.2.1.R4 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meng-upgrade

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.2.1.R3 (2019-12-12)

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug di mana `OSGP` telah dinonaktifkan meskipun fitur diaktifkan dengan benar menggunakan Mode Lab yang menggunakan nilai `ObjectIndex` dalam parameter `neptune_lab_mode`.
- Memperbaiki bug yang mempengaruhi kueri Gremlin dengan `.fold()` di dalam langkah `.project().by()`. Sebagai contoh, hal itu menyebabkan kueri berikut untuk mengembalikan hasil yang tidak lengkap:

```
g.V().project("a").by(valueMap().fold())
```

- Memperbaiki hambatan performa dalam pemuatan massal data RDF.
- Memperbaiki bug yang menyebabkan crash pada replika saat aliran diaktifkan dan replika dimulai ulang sebelum primer.
- Memperbaiki bug di mana sertifikat SSL yang diputar pada instans tidak diambil tanpa merestart instans.

Versi Bahasa Kueri yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.2.1.R3, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.1`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.2.1.R3

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Namun, pembaruan otomatis ke rilis ini tidak didukung.

Peningkatan ke Rilis Ini

Amazon Neptune 1.0.2.1.R3 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.2.1.R2 (2019-11-25)

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug yang mempengaruhi semua kueri `project().by()` dengan non round-robin by-traversals dan non `path()` by-traversals.

Versi Bahasa Kueri yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.2.1.R2, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.1`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.2.1.R2

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Namun, pembaruan otomatis ke rilis ini tidak didukung.

Peningkatan ke Rilis Ini

Amazon Neptune 1.0.2.1.R2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada peningkatan jalur untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.2.0 (2019-11-08)

PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN

Mulai dari 2020-05-19, tidak ada instance baru yang menggunakan versi mesin ini yang akan dibuat.

Versi mesin ini sekarang digantikan oleh versi 1.0.2.1, yang berisi semua perbaikan bug dalam versi ini serta fitur tambahan seperti integrasi pencarian teks lengkap, dukungan indeks OSGP, dan salinan cluster snapshot database di seluruh Wilayah. AWS

Mulai 1 Juni 2020, Neptune akan secara otomatis meng-upgrade klaster yang menjalankan versi mesin ini ke patch terbaru dari versi 1.0.2.1 selama jendela pemeliharaan berikutnya. Anda dapat meng-upgrade secara manual sebelum tanggal tersebut, seperti yang dijelaskan di sini.

Jika Anda memiliki masalah dengan upgradenya, silahkan hubungi kami melalui Dukungan AWS atau Forum Developer AWS.

Rilis Patch Berikutnya untuk Rilis Ini

- Rilis: 1.0.2.0.R3 (2020-05-05)

- Rilis: 1.0.2.0.R2 (2019-11-21)

Fitur Baru dalam Rilis Mesin Ini

Selain pembaruan pemeliharaan, rilis ini menambahkan fungsionalitas baru untuk mendukung lebih dari satu versi mesin sekaligus (lihat Mempertahankan Cluster DB Amazon Neptunus Anda).

Akibatnya, penomoran rilis mesin berubah (lihat Penomoran versi sebelum rilis mesin 1.3.0.0).

Versi Bahasa Kueri yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.2.0, pastikan bahwa proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.1`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.2.0

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Anda tidak akan secara otomatis meningkatkan ke rilis ini.

Peningkatan ke Rilis Ini

Amazon Neptune versi 1.0.2.0 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada jalur peningkatan untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.2.0.R3 (2020-05-05)

PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN

Mulai dari 2020-05-19, tidak ada instance baru yang menggunakan versi mesin ini yang akan dibuat.

Versi mesin ini sekarang digantikan oleh versi 1.0.2.1, yang berisi semua perbaikan bug dalam versi ini serta fitur tambahan seperti integrasi pencarian teks lengkap, dukungan indeks OSGP, dan salinan cluster snapshot database di seluruh Wilayah. AWS

Mulai 1 Juni 2020, Neptune akan secara otomatis meng-upgrade klaster yang menjalankan versi mesin ini ke patch terbaru dari versi 1.0.2.1 selama jendela pemeliharaan berikutnya. Anda dapat meng-upgrade secara manual sebelum tanggal tersebut, seperti yang dijelaskan di sini.

Jika Anda memiliki masalah dengan upgradenya, silahkan hubungi kami melalui Dukungan AWS atau Forum Developer AWS.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug di mana `ConcurrentModificationConflictException` dan `TransactionException` dilaporkan sebagai `InternalFailureException` generik.
- Memperbaiki bug dalam pemeriksaan kondisi yang menyebabkan seringnya restart server selama start up.
- Memperbaiki bug di mana data tidak terlihat pada replika karena komit rusak dalam kondisi tertentu.
- Memperbaiki bug dalam serialisasi status pemuatan di mana pemuatan gagal karena kurangnya izin akses Amazon S3.
- Memperbaiki kebocoran sumber daya dalam sesi Gremlin.
- Memperbaiki bug dalam pemeriksaan kondisi yang menyembunyikan status tidak sehat saat memulai komponen yang mengelola autentikasi IAM.
- Memperbaiki bug di mana Neptunus gagal mengirim WebSocket bingkai tutup sebelum menutup saluran.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.2.0.R3, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.1`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.2.0.R3

Klaster Anda akan ditingkatkan ke rilis patch ini secara otomatis selama jendela pemeliharaan berikutnya jika Anda menggunakan versi mesin `1.0.2.0`.

Anda dapat secara manual meng-upgrade rilis mesin Neptune versi lebih awal ke rilis ini.

Meningkatkan ke Rilis Ini

Mesin Amazon Neptune versi 1.0.2.0.R3 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada jalur peningkatan untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi

syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meng-upgrade

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus bergantung pada ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri, serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan untuk membiarkan pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.2.0.R2 (2019-11-21)

PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN

Mulai dari 2020-05-19, tidak ada instance baru yang menggunakan versi mesin ini yang akan dibuat.

Versi mesin ini sekarang digantikan oleh versi 1.0.2.1, yang berisi semua perbaikan bug dalam versi ini serta fitur tambahan seperti integrasi pencarian teks lengkap, dukungan indeks OSGP, dan salinan cluster snapshot database di seluruh Wilayah. AWS

Mulai 1 Juni 2020, Neptune akan secara otomatis meng-upgrade klaster yang menjalankan versi mesin ini ke patch terbaru dari versi 1.0.2.1 selama jendela pemeliharaan berikutnya. Anda dapat meng-upgrade secara manual sebelum tanggal tersebut, seperti yang dijelaskan di sini.

Jika Anda memiliki masalah dengan upgradenya, silahkan hubungi kami melalui Dukungan AWS atau Forum Developer AWS.

Perbaikan Cacat dalam Rilis Mesin Ini

- Meningkatkan strategi caching untuk halaman kotor di server sehingga `FreeableMemory` pulih lebih cepat ketika server memasuki keadaan memori rendah.
- Memperbaiki bug yang dapat menyebabkan kondisi balapan dan crash ketika banyak permintaan pemuatan and/or mulai status pemuatan bersamaan diproses di server.

Versi Kueri Bahasa yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.2.0.R2, pastikan bahwa proyek Anda kompatibel dengan versi kueri bahasa ini:

- Versi Gremlin: `3.4.1`
- Versi SPARQL: `1.1`

Jalur Peningkatan untuk Rilis Mesin 1.0.2.0.R2

Anda secara manual dapat meningkatkan rilis mesin Neptune apa pun sebelumnya ke rilis ini.

Namun, pembaruan otomatis ke rilis ini tidak didukung.

Peningkatan ke Rilis Ini

Mesin Amazon Neptune versi 1.0.2.0.R2 sekarang tersedia secara umum.

Jika klaster DB menjalankan versi mesin yang ada jalur peningkatan untuk rilis ini, klaster tersebut sekarang memenuhi syarat untuk ditingkatkan. Anda dapat meningkatkan klaster yang memenuhi syarat menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Perintah CLI berikut akan meningkatkan klaster yang memenuhi syarat dengan segera:

Untuk Linux, macOS, atau Unix:

```
aws neptune modify-db-cluster \
```

```
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Untuk Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan memerlukan basis data dimulai ulang di instans tersebut, sehingga Anda mengalami waktu henti 20 hingga 30 detik, setelah itu Anda dapat melanjutkan penggunaan klaster DB Anda.

Selalu uji sebelum Anda meningkatkan

Saat versi mesin Neptunus mayor atau minor baru dirilis, selalu uji aplikasi Neptunus Anda terlebih dahulu sebelum memutakhirkannya. Bahkan peningkatan kecil dapat memperkenalkan fitur atau perilaku baru yang akan memengaruhi kode Anda.

Mulailah dengan membandingkan halaman catatan rilis dari versi Anda saat ini dengan versi yang ditargetkan untuk melihat apakah akan ada perubahan dalam versi bahasa kueri atau perubahan melanggar lainnya.

Cara terbaik untuk menguji versi baru sebelum memutakhirkan cluster DB produksi Anda adalah dengan mengkloning cluster produksi Anda sehingga klon menjalankan versi mesin baru. Anda kemudian dapat menjalankan kueri pada klon tanpa mempengaruhi cluster DB produksi.

Selalu buat snapshot manual sebelum Anda meningkatkan

Sebelum melakukan upgrade, kami sangat menyarankan agar Anda selalu membuat snapshot manual dari cluster DB Anda. Memiliki snapshot otomatis hanya menawarkan perlindungan jangka pendek, sedangkan snapshot manual tetap tersedia sampai Anda menghapusnya secara eksplisit.

Dalam kasus tertentu Neptunus membuat snapshot manual untuk Anda sebagai bagian dari proses peningkatan, tetapi Anda tidak harus mengandalkan ini, dan harus membuat snapshot manual Anda sendiri dalam hal apa pun.

Ketika Anda yakin bahwa Anda tidak perlu mengembalikan cluster DB Anda ke status pra-pemutakhiran, Anda dapat secara eksplisit menghapus snapshot manual yang Anda buat sendiri,

serta snapshot manual yang mungkin dibuat Neptunus. Jika Neptunus membuat snapshot manual, itu akan memiliki nama yang dimulai `preupgrade` dengan, diikuti dengan nama cluster DB Anda, versi mesin sumber, versi mesin target, dan tanggal.

> **Note**
>
> Jika Anda mencoba memutakhirkan saat tindakan yang tertunda sedang dalam proses, Anda mungkin mengalami kesalahan seperti berikut:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Jika Anda mengalami kesalahan ini, tunggu hingga tindakan yang tertunda selesai, atau segera picu jendela pemeliharaan agar pemutakhiran sebelumnya selesai.

Untuk informasi selengkapnya tentang peningkatan versi mesin Anda, lihat Mempertahankan Cluster DB Amazon Neptunus Anda. Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Versi Mesin Amazon Neptune 1.0.1.2 (2020-06-10)

PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN

Mulai dari 2021-04-27, tidak ada instance baru yang menggunakan versi mesin ini yang akan dibuat.

Perbaikan dalam Rilis Mesin Ini

- Neptune sekarang memunculkan `IllegalArgumentException` jika Anda mencoba mengakses properti, vertex, atau edge yang tidak ada. Sebelumnya, Neptune memunculkan `UnsupportedOperationException` dalam situasi itu.

  Sebagai contoh, jika Anda mencoba menambahkan sebuah edge yang mereferensikan sebuah vertex yang tidak ada, Anda sekarang akan memunculkan `IllegalArgumentException`.

Perbaikan Cacat dalam Rilis Mesin Ini

- Perbaikan bug di mana komit kamus dan transaksi pengguna rusak ketika dua pemetaan `value->id` sedang dimasukkan secara bersamaan.
- Perbaikan bug dalam serialisasi status pemuatan.
- Perbaikan kegagalan stochastic di server startup menunda pembuatan instans.
- Perbaikan kebocoran kursor.

Versi Bahasa Kueri yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.1.2, pastikan bahwa proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.4.1`
- Versi SPARQL: `1.1`

Versi Mesin Amazon Neptune 1.0.1.1 (2020-06-26)

PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN

Mulai dari 2021-04-27, tidak ada instance baru yang menggunakan versi mesin ini yang akan dibuat.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug di mana komit telah rusak ketika dimasukkan secara bersamaan.
- Perbaikan bug dalam serialisasi status pemuatan.
- Perbaikan kegagalan stochastic di server startup menunda pembuatan instans.
- Memperbaiki kebocoran memori.

Versi Bahasa Kueri yang Didukung dalam Rilis Ini

Sebelum meningkatkan klaster DB ke versi 1.0.1.1, pastikan bahwa proyek Anda kompatibel dengan versi bahasa kueri ini:

- Versi Gremlin: `3.3.2`

- Versi SPARQL: `1.1`

Versi Mesin Amazon Neptune 1.0.1.0 (2019-07-02)

PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN

Mulai dari 2021-04-27, tidak ada instance baru yang menggunakan versi mesin ini yang akan dibuat.

Pembaruan Mesin Amazon Neptune 2019-10-31

Versi: 1.0.1.0.200502.0

PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN

Tidak ada instans baru yang menggunakan versi mesin ini akan dibuat, mulai 2021-04-27.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin dalam serialisasi respons langkah `tree()` saat klien terhubung ke Neptune menggunakan `traversal().withRemote(...)` (dengan kata lain, menggunakan GLV bytecode).

  Rilis ini membahas masalah di mana klien yang terhubung ke Neptune menggunakan `traversal().withRemote(...)` menerima respons yang tidak valid untuk kueri Gremlin yang berisi langkah `tree()`.
- Memperbaiki bug SPARQL di kueri `DELETE WHERE LIMIT`, di mana proses penghentian kueri akan menggantung karena kondisi balapan, menyebabkan kueri kehabisan waktu.

Pembaruan Mesin Amazon Neptune 2019-10-15

Versi: 1.0.1.0.200463.0

PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN

Tidak ada instans baru yang menggunakan versi mesin ini akan dibuat, mulai 2021-04-27.

Fitur Baru Dalam Rilis Mesin Ini

- Menambahkan Explain/Profile fitur Gremlin (lihatMenganalisis eksekusi kueri Neptune menggunakan `explain` Gremlin).

- Menambahkan Support untuk sesi berbasis skrip Gremlin untuk memungkinkan eksekusi beberapa traversal Gremlin dalam satu transaksi.
- Menambahkan dukungan untuk ekstensi Kueri Gabungan SPARQL di Neptune (lihat Kueri Gabungan SPARQL 1.1 dan Kueri gabungan SPARQL di Neptune menggunakan ekstensi `SERVICE`).
- Menambahkan fitur yang memungkinkan Anda menyuntikkan `queryId` Anda sendiri ke dalam kueri Gremlin atau SPARQL, baik melalui parameter URL HTTP atau melalui petunjuk kueri `queryId` SPARQL (lihat Menyuntikkan ID Kustom Ke Dalam Gremlin Neptune atau Kueri SPARQL).
- Menambahkan fitur Mode Lab ke Neptune yang dapat memungkinkan Anda untuk mencoba fitur-fitur mendatang yang belum siap untuk digunakan dalam produksi.
- Menambahkan fitur Aliran Neptunus mendatang yang andal mencatat setiap perubahan yang dibuat ke basis data Anda ke aliran yang bertahan selama seminggu. Fitur ini hanya tersedia di Mode Lab.
- Memperbarui semantik formal untuk transaksi bersamaan (lihat Semantik Transaksi di Neptune). Fitur ini memberikan jaminan standar industri sekitar konkurensi.

  Secara default, semantik transaksi ini diaktifkan. Dalam beberapa skenario, fitur ini dapat mengubah perilaku beban saat ini dan mengurangi performa beban. Anda dapat menggunakan parameter `neptune_lab_mode` Klaster DB untuk kembali ke semantik sebelumnya dengan memasukkan `ReadWriteConflictDetection=disabled` dalam nilai parameter.

Perbaikan dalam Rilis Mesin Ini

- Memperbaiki Status instans API dengan melaporkan versi apa TinkerPop dan versi SPARQL apa yang digunakan mesin.
- Peningkatan performa operator subgrafik Gremlin.
- Meningkatkan performa serialisasi respons Gremlin.
- Peningkatan performa dalam langkah Gremlin Union.
- Meningkatkan latensi kueri SPARQL sederhana.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug Gremlin dimana waktu habis salah dikembalikan sebagai kegagalan internal.

- Memperbaiki bug SPARQL di mana ORDER BY di atas sekumpulan variabel menyebabkan Kesalahan Server Internal.

Pembaruan Mesin Amazon Neptune 2019-09-19

PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN

Tidak ada instans baru yang menggunakan versi mesin ini akan dibuat, mulai 2021-04-27.

Versi: 1.0.1.0.200457.0

Amazon Neptune versi 1.0.1.0.200457.0 sekarang tersedia secara umum. Semua klaster DB Neptune baru, termasuk yang dipulihkan dari snapshot, akan dibuat di Neptune 1.0.1.0.200457.0 setelah pembaruan mesin selesai untuk Wilayah tersebut.

Klaster yang ada dapat ditingkatkan ke rilis ini dengan segera menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Anda dapat menggunakan perintah CLI berikut untuk meng-upgrade klaster DB:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan yang memerlukan basis data dimulai ulang di semua instans dalam klaster DB, sehingga Anda mengalami waktu henti 20-30 detik hingga beberapa menit, setelah itu Anda dapat melanjutkan penggunaan klaster atau instans DB Anda. Anda dapat melihat atau mengubah pengaturan jendela pemeliharaan dari konsol Neptune.

Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki masalah kebenaran Gremlin yang muncul di rilis mesin sebelumnya (1.0.1.0.200369.0) dengan menghapus perbaikan performa ke penanganan predikat konjungtif yang menyebabkannya.
- Memperbaiki bug SPARQL yang menyebabkan kueri dengan `DISTINCT` dan satu pola yang dibungkus menjadi `OPTIONAL` untuk menghasilkan `InternalServerError`.

Pembaruan Mesin Amazon Neptune 2019-08-13

PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN

Mulai dari 2021-04-27, tidak ada instance baru yang menggunakan versi mesin ini yang akan dibuat.

Fitur Baru dalam Rilis Mesin Ini

- Menambahkan opsi `OVERSUBSCRIBE` ke parameter `parallelism` dari Perintah Loader Neptune, yang menyebabkan loader massal Neptune menggunakan semua utas dan sumber daya yang tersedia.

Perbaikan dalam Rilis Mesin Ini

- Peningkatan performa filter SPARQL yang berisi ekspresi OR logika sederhana.
- Peningkatan performa Gremlin dalam menangani predikat konjungtif.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug SPARQL yang mencegah pengurangan `xsd:duration` dari `xsd:date`.
- Memperbaiki bug SPARQL yang menyebabkan hasil yang tidak lengkap dari inlining statis dalam keberadaan `UNION`.
- Memperbaiki bug SPARQL dalam pembatalan kueri.
- Memperbaiki bug Gremlin yang menyebabkan overflow selama promosi jenis.
- Memperbaiki bug Gremlin dalam penanganan elemen vertex di langkah `addE().from().to()`.
- Memperbaiki bug Gremlin (dirilis 2019-07-26 di Mesin versi 1.0.1.0.200366.0) yang melibatkan penanganan NaN ganda dan mengapung dalam sisipan kardinalitas tunggal.
- Memperbaiki bug dalam menghasilkan rencana kueri yang melibatkan pencarian berbasis properti.

Pembaruan Mesin Amazon Neptune 2019-07-26

Versi: 1.0.1.0.200366.0

PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN

Tidak ada instans baru yang menggunakan versi mesin ini akan dibuat, mulai 2021-04-27.

Fitur Baru Dalam Rilis Mesin Ini

- Diupgrade ke TinkerPop 3.4.1 (lihat TinkerPop Upgrade Informasi, dan TinkerPop3.4.1 Ubah Log).

  Untuk pelanggan Neptune, perubahan ini menyediakan fungsionalitas dan perbaikan baru, seperti:

  - `GraphBinary` sekarang tersedia sebagai format serialisasi.
  - Bug keep-alive yang menyebabkan kebocoran memori di driver TinkerPop Java telah diperbaiki, jadi solusi tidak lagi diperlukan.

  Namun, dalam beberapa kasus, mereka dapat mempengaruhi kode Gremlin yang ada di Neptune. Misalnya:

  - `valueMap()` sekarang mengembalikan `Map<Object,Object>` bukan `Map<String,Object>`.
  - Perilaku tidak konsisten langkah `within()` diperbaiki sehingga langkah bekerja secara konsisten dengan langkah-langkah lain. Sebelumnya, jenis harus cocok agar perbandingan bekerja. Sekarang, jumlah jenis yang berbeda dapat dibandingkan secara akurat. Misalnya, `33` sekarang membandingkan sebagai sama dengan `33L`, yang tidak pernah terjadi sebelumnya.
  - Sebuah bug di `ReducingBarrierStep` telah diperbaiki, sehingga sekarang tidak mengembalikan nilai jika tidak ada elemen yang tersedia untuk output.
  - Urutan cakupan `select()` berubah (urutan sekarang `maps`, `side-effects`, `paths`). Hal ini mengubah hasil kueri langka yang menggabungkan `side-effects` dan `select` dengan nama kunci yang sama untuk `side-effects` seperti untuk `select`.
  - `bulkSet()` sekarang menjadi bagian dari protokol GraphSon. Kueri yang berakhir dengan `toBulkSet()` tidak akan bekerja dengan klien lama.
  - Salah satu parameterisasi langkah `Submit()` telah dihapus dari klien 3.4.

  Banyak perubahan lain yang diperkenalkan di TinkerPop 3.4 tidak mempengaruhi perilaku Neptunus saat ini. Misalnya, `io()` Gremlin ditambahkan sebagai langkah ke `Traversal` dan sekarang tidak lagi digunakan dalam `Graph`, tetapi tidak pernah diaktifkan di Neptune.
- Menambahkan dukungan untuk sifat vertex kardinalitas tunggal ke loader massal untuk Gremlin, untuk memuat data grafik properti.
- Menambahkan opsi untuk menimpa nilai yang ada untuk properti kardinalitas tunggal di loader massal.
- Menambahkan kemampuan untuk mengambil status kueri Gremlin, dan untuk membatalkan kueri Gremlin.

- Menambahkan petunjuk kueri untuk batas waktu kueri SPARQL.
- Menambahkan kemampuan untuk melihat peran instans dalam API status (lihat Status instans).
- Menambahkan dukungan untuk kloning basis data (lihat Kloning Basis Data di Neptune).

Perbaikan dalam Rilis Mesin Ini

- Peningkatan Penjelasan Kueri SPARQL untuk menunjukkan variabel grafik dari klausa FROM.
- Peningkatan performa untuk SPARQL dalam filter, filter yang sama, klausa VALUES, dan jumlah rentang.
- Peningkatan performa untuk pengurutan langkah Gremlin.
- Peningkatan performa untuk traversal `.repeat.dedup` Gremlin.
- Peningkatan performa traversal `valueMap()` dan `path().by()` Gremlin.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki beberapa masalah dengan jalur properti SPARQL termasuk operasi dengan grafik bernama.
- Memperbaiki masalah dengan kueri SPARQL CONTRUCT yang menyebabkan masalah memori.
- Memperbaiki masalah dengan parser RDF Turtle dan nama lokal.
- Memperbaiki masalah untuk memperbaiki pesan kesalahan yang ditampilkan kepada pengguna.
- Memperbaiki masalah dengan traversal `repeat()...drop()` Gremlin.
- Memperbaiki masalah dengan langkah `drop()` Gremlin.
- Memperbaiki masalah dengan filter label Gremlin.
- Memperbaiki masalah dengan batas waktu kueri Gremlin.

Pembaruan Mesin Amazon Neptune 2019-07-02

PENTING: VERSI MESIN INI SEKARANG TIDAK LAGI DIGUNAKAN

Tidak ada instans baru yang menggunakan versi mesin ini akan dibuat, mulai 2021-04-27.

Perbaikan Cacat dalam Rilis Mesin Ini

- Memperbaiki bug yang menyebabkan pola tertentu dengan nama properti dan nilai terikat untuk tidak dioptimalkan.

Rilis Mesin Neptune Sebelumnya

Topik

- Pembaruan Mesin Amazon Neptune 2019-06-12
- Pembaruan Mesin Amazon Neptune 2019-05-01
- Pembaruan Mesin Amazon Neptune 2019-01-21
- Pembaruan Mesin Amazon Neptune 2018-11-19
- Pembaruan Mesin Amazon Neptune 2018-11-08
- Pembaruan Mesin Amazon Neptune 2018-10-29
- Pembaruan mesin Amazon Neptune 2018-09-06
- Pembaruan Mesin Amazon Neptune 2018-07-24
- Pembaruan Mesin Amazon Neptune 2018-06-22

Pembaruan Mesin Amazon Neptune 2019-06-12

Versi: 1.0.1.0.200310.0

Amazon Neptune versi 1.0.1.0.200310.0 sekarang tersedia secara umum. Semua klaster DB Neptune baru, termasuk yang dipulihkan dari snapshot, akan dibuat di Neptune 1.0.1.0.200310.0 setelah pembaruan mesin selesai untuk Wilayah tersebut.

Klaster yang ada dapat ditingkatkan ke rilis ini dengan segera menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Anda dapat menggunakan perintah CLI berikut untuk meng-upgrade klaster DB untuk rilis ini dengan segera:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Klaster Neptune DB akan secara otomatis ditingkatkan ke mesin rilis 1.0.1.0.200310.0 selama jendela pemeliharaan sistem. Pengaturan waktu saat pembaruan diterapkan tergantung pada Wilayah dan pengaturan jendela pemeliharaan untuk klaster DB, serta pada jenis pembaruan.

> **Note**
>
> Jendela pemeliharaan instans tidak berlaku untuk pembaruan mesin.

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan yang memerlukan basis data dimulai ulang di semua instans dalam klaster DB, sehingga Anda mengalami waktu henti 20-30 detik hingga beberapa menit, setelah itu Anda dapat melanjutkan penggunaan klaster atau instans DB Anda. Anda dapat melihat atau mengubah pengaturan jendela pemeliharaan dari konsol Neptune.

Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Perbaikan

- Memperbaiki bug di mana penyisipan bersamaan dan menjatuhkan edge dapat mengakibatkan beberapa edge dengan id yang sama.
- Perbaikan dan peningkatan kecil lainnya.

Pembaruan Mesin Amazon Neptune 2019-05-01

Versi: 1.0.1.0.200296.0

Amazon Neptune versi 1.0.1.0.200296.0 sekarang tersedia secara umum. Semua klaster DB Neptune baru, termasuk yang dipulihkan dari snapshot, akan dibuat di Neptune 1.0.1.0.200296.0 setelah pembaruan mesin selesai untuk Wilayah tersebut.

Klaster yang ada dapat ditingkatkan ke rilis ini dengan segera menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Anda dapat menggunakan perintah CLI berikut untuk meng-upgrade klaster DB untuk rilis ini dengan segera:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
```

```
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Klaster Neptune DB akan secara otomatis ditingkatkan ke mesin rilis 1.0.1.0.200296.0 selama jendela pemeliharaan sistem. Pengaturan waktu saat pembaruan diterapkan tergantung pada Wilayah dan pengaturan jendela pemeliharaan untuk klaster DB, serta pada jenis pembaruan.

> **Note**
>
> Jendela pemeliharaan instans tidak berlaku untuk pembaruan mesin.

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan yang memerlukan basis data dimulai ulang di semua instans dalam klaster DB, sehingga Anda mengalami waktu henti 20-30 detik hingga beberapa menit, setelah itu Anda dapat melanjutkan penggunaan klaster atau instans DB Anda. Anda dapat melihat atau mengubah pengaturan jendela pemeliharaan dari konsol Neptune.

Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Perbaikan

- Menambahkan fitur `explain` baru ke kueri SPARQL Neptune untuk membantu Anda memvisualisasikan rencana kueri dan mengambil langkah-langkah untuk mengoptimalkan jika diperlukan. Untuk informasi, lihat `explain` SPARQL.
- Peningkatan performa dan pelaporan SPARQL dengan berbagai cara.
- Peningkatan performa dan perilaku Gremlin dalam berbagai cara.
- Meningkatkan batas waktu dari kueri `drop( )` jangka panjang.
- Peningkatan performa kueri `otherV( )`.
- Menambahkan dua bidang untuk informasi yang dikembalikan ketika Anda mengkueri status kesehatan Neptune klaster atau instans DB, yaitu nomor versi mesin dan waktu mulai klaster atau instans. Lihat Status instans.
- API `Get-Status` loaded Neptune sekarang mengembalikan bidang `startTime` yang mencatat ketika tugas pemuatan dimulai.
- Perintah loader sekarang mengambil parameter `parallelism` pilihan yang memungkinkan Anda membatasi jumlah utas yang digunakan loader.

Pembaruan Mesin Amazon Neptune 2019-01-21

Versi: 1.0.1.0.200267.0

Amazon Neptune versi 1.0.1.0.200267.0 sekarang tersedia secara umum. Semua klaster DB Neptune baru, termasuk yang dipulihkan dari snapshot, akan dibuat di Neptune 1.0.1.0.200267.0 setelah pembaruan mesin selesai untuk Wilayah tersebut.

Klaster yang ada dapat ditingkatkan ke rilis ini dengan segera menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Anda dapat menggunakan perintah CLI berikut untuk meng-upgrade klaster DB untuk rilis ini dengan segera:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Klaster Neptune DB akan secara otomatis ditingkatkan ke mesin rilis 1.0.1.0.200267.0 selama jendela pemeliharaan sistem. Pengaturan waktu saat pembaruan diterapkan tergantung pada Wilayah dan pengaturan jendela pemeliharaan untuk klaster DB, serta pada jenis pembaruan.

> **Note**
>
> Jendela pemeliharaan instans tidak berlaku untuk pembaruan mesin.

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan yang memerlukan basis data dimulai ulang di semua instans dalam klaster DB, sehingga Anda mengalami waktu henti 20-30 detik hingga beberapa menit, setelah itu Anda dapat melanjutkan penggunaan klaster atau instans DB Anda. Anda dapat melihat atau mengubah pengaturan jendela pemeliharaan dari konsol Neptune.

Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Perbaikan

- Neptune menunggu lebih lama (dalam batas waktu kueri yang ditentukan) untuk menyelesaikan setiap konflik. Hal ini mengurangi jumlah pengecualian modifikasi bersamaan yang perlu ditangani oleh klien (lihat Kesalahan Kueri).

- Memperbaiki masalah saat penegakan kardinalitas Gremlin terkadang menyebabkan mesin di-restart.
- Peningkatan performa Gremlin untuk kueri berulang `emit.times`.
- Memperbaiki masalah Gremlin di mana `repeat.until` yang memungkinkan solusi `.emit` lewat yang seharusnya telah disaring.
- Peningkatan penanganan kesalahan di Gremlin.

Pembaruan Mesin Amazon Neptune 2018-11-19

Versi: 1.0.1.0.200264.0

Amazon Neptune versi 1.0.1.0.200264.0 sekarang tersedia secara umum. Semua klaster DB Neptune baru, termasuk yang dipulihkan dari snapshot, akan dibuat di Neptune 1.0.1.0.200264.0 setelah pembaruan mesin selesai untuk Wilayah tersebut.

Klaster yang ada dapat ditingkatkan ke rilis ini dengan segera menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Anda dapat menggunakan perintah CLI berikut untuk meng-upgrade klaster DB untuk rilis ini dengan segera:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Klaster Neptune DB akan secara otomatis ditingkatkan ke mesin rilis 1.0.1.0.200264.0 selama jendela pemeliharaan sistem. Pengaturan waktu saat pembaruan diterapkan tergantung pada Wilayah dan pengaturan jendela pemeliharaan untuk klaster DB, serta pada jenis pembaruan.

> **Note**
>
> Jendela pemeliharaan instans tidak berlaku untuk pembaruan mesin.

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan yang memerlukan basis data dimulai ulang di semua instans dalam klaster DB, sehingga Anda mengalami waktu henti 20-30 detik hingga beberapa menit, setelah itu Anda dapat melanjutkan penggunaan klaster atau instans DB Anda. Anda dapat melihat atau mengubah pengaturan jendela pemeliharaan dari konsol Neptune.

Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Perbaikan

- menambahkan dukungan untuk the section called “Petunjuk kueri”.
- Peningkatan pesan kesalahan untuk autentikasi IAM. Untuk informasi selengkapnya, lihat the section called “Kesalahan IAM”.
- Peningkatan performa kueri SPARQL dengan jumlah predikat yang tinggi.
- Peningkatan performa jalur properti SPARQL.
- Peningkatan performa Gremlin untuk mutasi bersyarat, seperti `fold().coalesce(unfold(), …)`, bila digunakan dengan langkah `addV()`, `addE()`, dan `property()`.
- Peningkatan performa Gremlin untuk modulasi `by()` dan `sack()`.
- Peningkatan performa Gremlin untuk langkah `group()` dan `groupCount()`.
- Peningkatan performa Gremlin untuk langkah `store()`, `sideEffect()`, dan `cap().unfold()`.
- Peningkatan dukungan untuk kendala properti kardinalitas tunggal Gremlin.
  - Peningkatan penegakan kardinalitas tunggal untuk properti edge dan properti vertex yang ditandai sebagai sifat kardinalitas tunggal.
  - Menunjukkan kesalahan jika nilai properti tambahan ditentukan untuk properti edge yang ada selama tugas Pemuatan Neptune.

Pembaruan Mesin Amazon Neptune 2018-11-08

Versi: 1.0.1.0.200258.0

Amazon Neptune versi 1.0.1.0.200258.0 sekarang tersedia secara umum. Semua klaster DB Neptune baru, termasuk yang dipulihkan dari snapshot, akan dibuat di Neptune 1.0.1.0.200258.0 setelah pembaruan mesin selesai untuk Wilayah tersebut.

Klaster yang ada dapat ditingkatkan ke rilis ini dengan segera menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Anda dapat menggunakan perintah CLI berikut untuk meng-upgrade klaster DB untuk rilis ini dengan segera:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
```

```
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Klaster Neptune DB akan secara otomatis ditingkatkan ke mesin rilis 1.0.1.0.200258.0 selama jendela pemeliharaan sistem. Pengaturan waktu saat pembaruan diterapkan tergantung pada Wilayah dan pengaturan jendela pemeliharaan untuk klaster DB, serta pada jenis pembaruan.

> **Note**
>
> Jendela pemeliharaan instans tidak berlaku untuk pembaruan mesin.

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan yang memerlukan basis data dimulai ulang di semua instans dalam klaster DB, sehingga Anda mengalami waktu henti 20-30 detik hingga beberapa menit, setelah itu Anda dapat melanjutkan penggunaan klaster atau instans DB Anda. Anda dapat melihat atau mengubah pengaturan jendela pemeliharaan dari konsol Neptune.

Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Perbaikan

- menambahkan dukungan untuk Petunjuk kueri SPARQL.
- Peningkatan performa untuk kueri SPARQL FILTER (NOT) Exists.
- Peningkatan performa untuk kueri SPARQL DESCRIBE.
- Peningkatan performa untuk pola repeat until di Gremlin.
- Peningkatan performa untuk menambahkan edge di Gremlin.
- Memperbaiki masalah di mana kueri SPARQL Update DELETE dapat gagal dalam beberapa kasus.
- Memperbaiki masalah untuk menangani batas waktu dengan server Gremlin WebSocket.

Pembaruan Mesin Amazon Neptune 2018-10-29

Versi: 1.0.1.0.200255.0

Amazon Neptune versi 1.0.1.0.200255.0 sekarang tersedia secara umum. Semua klaster DB Neptune baru, termasuk yang dipulihkan dari snapshot, akan dibuat di Neptune 1.0.1.0.200255.0 setelah pembaruan mesin selesai untuk Wilayah tersebut.

Klaster yang ada dapat ditingkatkan ke rilis ini dengan segera menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Anda dapat menggunakan perintah CLI berikut untuk meng-upgrade klaster DB untuk rilis ini dengan segera:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Klaster Neptune DB akan secara otomatis ditingkatkan ke mesin rilis 1.0.1.0.200255.0 selama jendela pemeliharaan sistem. Pengaturan waktu saat pembaruan diterapkan tergantung pada Wilayah dan pengaturan jendela pemeliharaan untuk klaster DB, serta pada jenis pembaruan.

> **Note**
>
> Jendela pemeliharaan instans tidak berlaku untuk pembaruan mesin.

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan yang memerlukan basis data dimulai ulang di semua instans dalam klaster DB, sehingga Anda mengalami waktu henti 20-30 detik hingga beberapa menit, setelah itu Anda dapat melanjutkan penggunaan klaster atau instans DB Anda. Anda dapat melihat atau mengubah pengaturan jendela pemeliharaan dari konsol Neptune.

Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Perbaikan

- Menambahkan informasi autentikasi IAM ke log Audit.
- Menambahkan Support untuk kredensial sementara menggunakan IAM role dan Profil Instans.
- Menambahkan penghentian WebSocket koneksi untuk otentikasi IAM saat izin dicabut atau jika pengguna atau peran IAM dihapus.
- Membatasi jumlah maksimum WebSocket koneksi hingga 60.000 per instance.
- Peningkatan performa Pemuatan Massal untuk tipe instans yang lebih kecil.
- Peningkatan performa untuk kueri yang menyertakan operator `and()`, `or()`, `not()`, `drop()` di Gremlin.
- NTriples Parser sekarang menolak tidak valid, seperti berisi spasi URIs putih. URIs

Pembaruan mesin Amazon Neptune 2018-09-06

Versi: 1.0.1.0.200237.0

Amazon Neptune versi 1.0.1.0.200237.0 sekarang tersedia secara umum. Semua klaster DB Neptune baru, termasuk yang dipulihkan dari snapshot, akan dibuat di Neptune 1.0.1.0.200237.0 setelah pembaruan mesin selesai untuk Wilayah tersebut.

Klaster yang ada dapat ditingkatkan ke rilis ini dengan segera menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Anda dapat menggunakan perintah CLI berikut untuk meng-upgrade klaster DB untuk rilis ini dengan segera:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Klaster Neptune DB akan secara otomatis ditingkatkan ke mesin rilis 1.0.1.0.200237.0 selama jendela pemeliharaan sistem. Pengaturan waktu saat pembaruan diterapkan tergantung pada Wilayah dan pengaturan jendela pemeliharaan untuk klaster DB, serta pada jenis pembaruan.

> **Note**
>
> Jendela pemeliharaan instans tidak berlaku untuk pembaruan mesin.

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan yang memerlukan basis data dimulai ulang di semua instans dalam klaster DB, sehingga Anda mengalami waktu henti 20-30 detik hingga beberapa menit, setelah itu Anda dapat melanjutkan penggunaan klaster atau instans DB Anda. Anda dapat melihat atau mengubah pengaturan jendela pemeliharaan dari konsol Neptune.

Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Perbaikan

- Memperbaiki masalah di mana beberapa kueri `SPARQL COUNT(DISTINCT)` gagal.
- Memperbaiki masalah di mana kueri `COUNT`, `SUM`, `MIN` dengan `DISTINCT` akan kehabisan memori.

- Memperbaiki masalah di mana data jenis `BLOB` akan menyebabkan pekerjaan Neptune Loader gagal.
- Memperbaiki masalah di mana sisipan duplikat akan menyebabkan kegagalan transaksi.
- Memperbaiki masalah di mana kueri `DROP ALL` tidak dapat dibatalkan.
- Memperbaiki masalah di mana klien Gremlin dapat hang sebentar-sebentar.
- Memperbarui semua kode kesalahan untuk muatan lebih besar dari 150M menjadi `HTTP 400`.
- Peningkatan kinerja dan akurasi single-triple-pattern `COUNT()` kueri.
- Peningkatan performa kueri `SPARQL UNION` dengan klausa `BIND`.

Pembaruan Mesin Amazon Neptune 2018-07-24

Versi: 1.0.1.0.200236.0

Amazon Neptune versi 1.0.1.0.200236.0 sekarang tersedia secara umum. Semua klaster DB Neptune baru, termasuk yang dipulihkan dari snapshot, akan dibuat di Neptune 1.0.1.0.200236.0 setelah pembaruan mesin selesai untuk Wilayah tersebut.

Klaster yang ada dapat ditingkatkan ke rilis ini dengan segera menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Anda dapat menggunakan perintah CLI berikut untuk meng-upgrade klaster DB untuk rilis ini dengan segera:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Klaster Neptune DB akan secara otomatis ditingkatkan ke mesin rilis 1.0.1.0.200236.0 selama jendela pemeliharaan sistem. Pengaturan waktu saat pembaruan diterapkan tergantung pada Wilayah dan pengaturan jendela pemeliharaan untuk klaster DB, serta pada jenis pembaruan.

> **Note**
>
> Jendela pemeliharaan instans tidak berlaku untuk pembaruan mesin.

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan yang memerlukan basis data dimulai ulang di semua instans dalam klaster DB, sehingga Anda mengalami

waktu henti 20-30 detik hingga beberapa menit, setelah itu Anda dapat melanjutkan penggunaan klaster atau instans DB Anda. Anda dapat melihat atau mengubah pengaturan jendela pemeliharaan dari konsol Neptune.

Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Perbaikan

- Memperbarui serialisasi SPARQL untuk `xsd:string` datatype. xsd:string tidak lagi disertakan dalam serialisasi JSON, yang sekarang konsisten dengan format output lainnya.
- Memperbaiki penanganan infinity `xsd:double`/`xsd:float`. Nilai `-INF`, `NaN`, dan `INF` sekarang diakui dan ditangani dengan benar dalam semua format loader data SPARQL, SPARQL 1.1 UPDATE, dan Kueri SPARQL 1.1.
- Memperbaiki masalah saat kueri Gremlin dengan nilai string kosong gagal secara tiba-tiba.
- Memperbaiki masalah saat `aggregate()` dan `cap()` Gremlin pada grafik kosong gagal tiba-tiba.
- Memperbaiki masalah di mana respons kesalahan salah dikembalikan untuk Gremlin ketika spesifikasi kardinalitas tidak valid, mis. `.property(set,id,'10')` dan `.property(single,id,'10')`.
- Memperbaiki masalah saat sintaks Gremlin yang tidak valid dikembalikan sebagai. InternalFailureException
- Memperbaiki ejaan di `TimeLimitExceeededException` ke `TimeLimitExceededException`, dalam pesan kesalahan.
- Mengubah titik akhir SPARQL dan GREMLIN merespon dengan cara yang konsisten ketika script tidak disediakan.
- Mengklarifikasi pesan kesalahan untuk terlalu banyak permintaan bersamaan.

Pembaruan Mesin Amazon Neptune 2018-06-22

Versi: 1.0.1.0.200233.0

Amazon Neptune versi 1.0.1.0.200233.0 sekarang tersedia secara umum. Semua klaster DB Neptune baru, termasuk yang dipulihkan dari snapshot, akan dibuat di Neptune 1.0.1.0.200233.0 setelah pembaruan mesin selesai untuk Wilayah tersebut.

Klaster yang ada dapat ditingkatkan ke rilis ini dengan segera menggunakan operasi klaster DB pada konsol atau dengan menggunakan SDK. Anda dapat menggunakan perintah CLI berikut untuk meng-upgrade klaster DB untuk rilis ini dengan segera:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Klaster Neptune DB akan secara otomatis ditingkatkan ke mesin rilis 1.0.1.0.200233.0 selama jendela pemeliharaan sistem. Pengaturan waktu saat pembaruan diterapkan tergantung pada Wilayah dan pengaturan jendela pemeliharaan untuk klaster DB, serta pada jenis pembaruan.

> **Note**
>
> Jendela pemeliharaan instans tidak berlaku untuk pembaruan mesin.

Pembaruan diterapkan ke semua instans dalam klaster DB pada saat yang sama. Pembaruan yang memerlukan basis data dimulai ulang di semua instans dalam klaster DB, sehingga Anda mengalami waktu henti 20-30 detik hingga beberapa menit, setelah itu Anda dapat melanjutkan penggunaan klaster atau instans DB Anda. Anda dapat melihat atau mengubah pengaturan jendela pemeliharaan dari konsol Neptune.

Jika Anda memiliki pertanyaan atau masalah, tim AWS Support tersedia di forum komunitas dan melalui AWS Premium Support.

Perbaikan

- Memperbaiki masalah saat sejumlah besar permintaan pemuatan massal dikeluarkan dalam hasil suksesi cepat dengan kesalahan.
- Memperbaiki masalah yang bergantung pada data di mana kueri bisa gagal dengan file. InternalServerError Contoh berikut menunjukkan jenis kueri yang terpengaruh.

```
g.V("my-id123").as("start").outE("knows").has("edgePropertyKey1",
 P.gt(0)).as("myedge").inV()
              .as("end").select("start", "end", "myedge").by("vertexPropertyKey1")
              .by("vertexPropertyKey1").by("edgePropertyKey1")
```

- Memperbaiki masalah di mana klien Gremlin Java tidak dapat terhubung ke server menggunakan WebSocket koneksi yang sama setelah batas waktu kueri yang berjalan lama.
- Memperbaiki masalah saat urutan yang diloloskan yang terkandung sebagai bagian dari kueri Gremlin melalui HTTP atau kueri berbasis string melalui koneksi tidak ditangani dengan benar. WebSocket

Pengantar menggunakan Amazon Neptunus APIs

APIs Manajemen Amazon Neptunus menyediakan dukungan SDK untuk membuat, mengelola, dan menghapus cluster dan instans Neptunus DB, sementara data Neptunus memberikan dukungan SDK untuk memuat data ke dalam grafik Anda, menjalankan APIs kueri, mendapatkan informasi tentang data dalam grafik Anda, dan menjalankan operasi pembelajaran mesin. Ini APIs tersedia dalam semua bahasa SDK. Dengan secara otomatis menandatangani permintaan API, mereka sangat mudah mengintegrasikan Neptunus ke dalam aplikasi.

Halaman ini memberikan informasi tentang cara menggunakannya APIs.

Tindakan IAM dengan nama yang berbeda dari rekan SDK API data Neptunus mereka

Saat memanggil metode API Neptunus di klaster yang mengaktifkan autentikasi IAM, Anda harus memiliki kebijakan IAM yang dilampirkan ke pengguna atau peran yang membuat panggilan yang memberikan izin untuk tindakan yang ingin Anda lakukan. Anda menetapkan izin tersebut dalam kebijakan menggunakan Tindakan IAM terkait. Anda juga dapat membatasi tindakan yang dapat diambil menggunakan kunci Kondisi IAM.

Sebagian besar tindakan IAM memiliki nama yang sama dengan metode API yang sesuai dengannya, tetapi beberapa metode dalam API data memiliki nama yang berbeda, karena beberapa dibagikan oleh lebih dari satu metode. Tabel di bawah ini mencantumkan metode data dan tindakan IAM yang sesuai:

Nama operasi API data	Korespondensi IAM
CancelGremlinQuery(cancel_gremlin_query)	Tindakan: `neptune-db: CancelQuery`
CancelLoaderJob(cancel_loader_job)	Tindakan: `neptune-db:CancelLoaderJob`
Batalkan MLData ProcessingJob (cancel_ml_data_processing_job)	Tindakan: `neptune-db:CancelMLDataProcessingJob`

Nama operasi API data	Korespondensi IAM
Batalkan MLModel TrainingJob (cancel_ml_model_training_job)	Tindakan: `neptune-db:CancelMLModelTrainingJob`
CancelOpenCypherQuery(cancel_open_cypher_query)	Tindakan: `neptune-db:` **`CancelQuery`**
Buat MLEndpoint (create_ml_endpoint)	Tindakan: `neptune-db:CreateMLEndpoint`
Hapus MLEndpoint (delete_ml_endpoint)	Tindakan: `neptune-db:DeleteMLEndpoint`
DeletePropertygraphStatistics(delete_propertygraph_statistics)	Tindakan: `neptune-db:` **`DeleteStatistics`**
DeleteSparqlStatistics(delete_sparql_statistics)	Tindakan: `neptune-db:` **`DeleteStatistics`**
ExecuteFastResetexecute_fast_reset ()	Tindakan: `neptune-db:` **`ResetDatabase`**
ExecuteGremlinExplainQuery(execute_gremlin_explain_query)	Tindakan: • `neptune-db:` **`ReadDataViaQuery`** • `neptune-db:` **`WriteDataViaQuery`** • `neptune-db:` **`DeleteDataViaQuery`** Kunci kondisi: `neptune-db:QueryLanguage:Gremlin`

Nama operasi API data	Korespondensi IAM
ExecuteGremlinProfileQuery(execute_gremlin_profile_query)	Tindakan: `neptune-db:` **`ReadDataViaQuery`** Kunci kondisi: `neptune-db:QueryLanguage:Gremlin`
ExecuteGremlinQuery(execute_gremlin_query)	Tindakan: • `neptune-db:` **`ReadDataViaQuery`** • `neptune-db:` **`WriteDataViaQuery`** • `neptune-db:` **`DeleteDataViaQuery`** Kunci kondisi: `neptune-db:QueryLanguage:Gremlin`
ExecuteOpenCypherExplainQuery(execute_open_cypher_explain_query)	Tindakan: `neptune-db:` **`ReadDataViaQuery`** Kunci kondisi: `neptune-db:QueryLanguage:OpenCypher`

Nama operasi API data	Korespondensi IAM	
ExecuteOpenCypherQuery(execute_open_cypher_query)	Tindakan: • `neptune-db: ReadDataViaQuery` • `neptune-db: WriteDataViaQuery` • `neptune-db: DeleteDataViaQuery` Kunci kondisi: `neptune-db:QueryLanguage:OpenCypher`	
GetEngineStatus(get_engine_status)	Tindakan: `neptune-db:GetEngineStatus`	
GetGremlinQueryStatus(get_gremlin_query_status)	Tindakan: `neptune-db: :GetQueryStatus` Kunci kondisi: `neptune-db:QueryLanguage:Gremlin`	
GetLoaderJobStatus(get_loader_job_status)	Tindakan: `neptune-db:GetLoaderJobStatus`	
Dapatkan MLData ProcessingJob (get_ml_data_processing_job)	Tindakan: `neptune-db: GetMLDataProcessingJobStatus`	
Dapatkan MLEndpoint (get_ml_endpoint)	Tindakan: `neptune-db: GetMLEndpointStatus`	

Nama operasi API data	Korespondensi IAM	
Dapatkan MLModel TrainingJob (get_ml_model_training_job)	Tindakan: `neptune-db:` **`GetMLModelTrainingJobStatus`**	
Dapatkan MLModel Transform Job (get_ml_model_transform_job)	Tindakan: `neptune-db:` **`GetMLModelTransformJobStatus`**	
GetOpenCypherQueryStatus(get_open_cypher_query_status)	Tindakan: `neptune-db:` **`:GetQueryStatus`** Kunci kondisi: `neptune-db:QueryLanguage:OpenCypher`	
GetPropertygraphStatistics(get_propertygraph_statistics)	Tindakan: `neptune-db:` **`GetStatisticsStatus`**	
GetPropertygraphStream(get_propertygraph_stream)	Tindakan: `neptune-db:` **`GetStreamRecords`** Kunci kondisi: • `neptune-db:QueryLanguage:Gremlin` • `neptune-db:QueryLanguage:OpenCypher`	
GetPropertygraphSummary(get_propertygraph_summary)	Tindakan: `neptune-db:` **`GetGraphSummary`**	
Dapatkan RDFGraph Ringkasan (get_rdf_graph_summary)	Tindakan: `neptune-db:` **`GetGraphSummary`**	

Nama operasi API data	Korespondensi IAM	
GetSparqlStatistics(get_sparql_statistics)	Tindakan: `neptune-db: GetStatisticsStatus`	
GetSparqlStream(get_sparql_stream)	Tindakan: `neptune-db: :GetStreamRecords` Kunci kondisi: `neptune-db:QueryLanguage:Sparql`	
ListGremlinQueries(list_gremlin_queries)	Tindakan: `neptune-db: :GetQueryStatus` Kunci kondisi: `neptune-db:QueryLanguage:Gremlin`	
Daftar MLEndpoints (list_ml_endpoints)	Tindakan: `neptune-db:ListMLEndpoints`	
Daftar MLModel TrainingJobs (list_ml_model_training_jobs)	Tindakan: `neptune-db:ListMLModelTrainingJobs`	
Daftar MLModel Transform Jobs (list_ml_model_transform_jobs)	Tindakan: `neptune-db:ListMLModelTransformJobs`	
ListOpenCypherQueries(list_open_cypher_queries)	Tindakan: `neptune-db: :GetQueryStatus` Kunci kondisi: `neptune-db:QueryLanguage:OpenCypher`	

Nama operasi API data	Korespondensi IAM
ManagePropertygraphStatistics(manage_propertygraph_statistics)	Tindakan: `neptune-db:` **`ManageStatistics`**
ManageSparqlStatistics(manage_sparql_statistics)	Tindakan: `neptune-db:` **`ManageStatistics`**
StartLoaderJob(start_loader_job)	Tindakan: `neptune-db:StartLoaderJob`
Mulai MLModel DataProcessingJob (start_ml_data_processing_job)	Tindakan: `neptune-db:StartMLModelDataProcessingJob`
Mulai MLModel TrainingJob (start_ml_model_training_job)	Tindakan: `neptune-db:StartMLModelTrainingJob`
Mulai MLModel TransformJob (start_ml_model_transform_job)	Tindakan: `neptune-db:StartMLModelTransformJob`

Referensi API Manajemen Amazon Neptunus

Bab ini mendokumentasikan operasi API Neptunus yang dapat Anda gunakan untuk mengelola dan memelihara kluster DB Neptunus Anda.

Neptune beroperasi di klaster server basis data yang terhubung dalam topologi replikasi. Dengan demikian, mengelola Neptunus sering melibatkan penerapan perubahan ke beberapa server dan memastikan bahwa semua replika Neptunus mengikuti server utama.

Karena Neptune secara transparan menskalakan penyimpanan yang mendasari saat data Anda tumbuh, pengelolaan Neptune memerlukan manajemen penyimpanan disk yang relatif kecil. Demikian pula, karena Neptune secara otomatis melakukan pencadangan berkelanjutan, klaster Neptune tidak memerlukan perencanaan atau waktu henti yang ekstensif untuk melakukan pencadangan.

Daftar Isi

- API Klaster DB Neptune
  - Buat DBCluster (tindakan)
  - Hapus DBCluster (tindakan)
  - Memodifikasi DBCluster (tindakan)
  - Mulai DBCluster (tindakan)
  - Berhenti DBCluster (tindakan)
  - AddRoleToDBCluster (tindakan)
  - RemoveRoleFromDBCluster (tindakan)
  - Failover DBCluster (tindakan)
  - PromoteReadReplicaDBCluster (tindakan)
  - Jelaskan DBClusters (tindakan)
  - Struktur:
  - DBCluster (struktur)
  - DBClusterAnggota (struktur)
  - DBClusterPeran (struktur)
  - CloudwatchLogsExportConfiguration (struktur)
  - PendingCloudwatchLogsExports (struktur)

- ClusterPendingModifiedValues (struktur)
- API Database Global Neptunus
  - CreateGlobalCluster (tindakan)
  - DeleteGlobalCluster (tindakan)
  - ModifyGlobalCluster (tindakan)
  - DescribeGlobalClusters (tindakan)
  - FailoverGlobalCluster (tindakan)
  - RemoveFromGlobalCluster (tindakan)
  - Struktur:
  - GlobalCluster (struktur)
  - GlobalClusterMember (struktur)
- API Instans Neptune
  - Buat DBInstance (tindakan)
  - Hapus DBInstance (tindakan)
  - Memodifikasi DBInstance (tindakan)
  - Reboot DBInstance (tindakan)
  - Jelaskan DBInstances (tindakan)
  - DescribeOrderableDBInstanceOpsi (tindakan)
  - DescribeValidDBInstanceModifikasi (tindakan)
  - Struktur:
  - DBInstance (struktur)
  - DBInstanceStatusInfo (struktur)
  - DBInstanceOpsi yang Dapat Dipesan (struktur)
  - PendingModifiedValues (struktur)
  - ValidStorageOptions (struktur)
  - Valid DBInstance ModificationsMessage (struktur)
- API Parameter Neptune
  - Salin DBParameter Grup (tindakan)
  - Salin DBCluster ParameterGroup (tindakan)
  - Buat DBParameter Grup (aksi)

- Buat DBCluster ParameterGroup (tindakan)
- Hapus DBParameter Grup (tindakan)
- Hapus DBCluster ParameterGroup (tindakan)
- Ubah DBParameter Grup (tindakan)
- Memodifikasi DBCluster ParameterGroup (tindakan)
- Atur Ulang DBParameter Grup (tindakan)
- Atur ulang DBCluster ParameterGroup (tindakan)
- Jelaskan DBParameters (tindakan)
- Jelaskan DBParameter Grup (tindakan)
- Jelaskan DBCluster Parameter (tindakan)
- Jelaskan DBCluster ParameterGroups (tindakan)
- DescribeEngineDefaultParameters (tindakan)
- DescribeEngineDefaultClusterParameters (tindakan)
- Struktur:
- Parameter (struktur)
- DBParameterKelompok (struktur)
- DBClusterParameterGroup (struktur)
- DBParameterGroupStatus (struktur)

- API Subnet Neptune
  - Buat DBSubnet Grup (aksi)
  - Hapus DBSubnet Grup (tindakan)
  - Ubah DBSubnet Grup (tindakan)
  - Jelaskan DBSubnet Grup (tindakan)
  - Struktur:
  - Subnet (struktur)
  - DBSubnetKelompok (struktur)
- API Snapshot Neptune
  - Buat DBCluster Snapshot (aksi)
  - Hapus DBCluster Snapshot (tindakan)
  - Salin DBCluster Snapshot (tindakan)

- Memodifikasi DBCluster SnapshotAttribute (tindakan)
- Kembalikan DBCluster FromSnapshot (tindakan)
- Kembalikan DBCluster ToPointInTime (tindakan)
- Jelaskan DBCluster Snapshots (tindakan)
- Jelaskan DBCluster SnapshotAttributes (tindakan)
- Struktur:
- DBClusterSnapshot (struktur)
- DBClusterSnapshotAttribute (struktur)
- DBClusterSnapshotAttributesResult (struktur)

- API Peristiwa Neptune
  - CreateEventSubscription (tindakan)
  - DeleteEventSubscription (tindakan)
  - ModifyEventSubscription (tindakan)
  - DescribeEventSubscriptions (tindakan)
  - AddSourceIdentifierToSubscription (tindakan)
  - RemoveSourceIdentifierFromSubscription (tindakan)
  - DescribeEvents (tindakan)
  - DescribeEventCategories (tindakan)
  - Struktur:
  - Peristiwa (struktur)
  - EventCategoriesMap (struktur)
  - EventSubscription (struktur)
- Neptunus lainnya APIs
  - AddTagsToResource (tindakan)
  - ListTagsForResource (tindakan)
  - RemoveTagsFromResource (tindakan)
  - ApplyPendingMaintenanceAction (tindakan)
  - DescribePendingMaintenanceActions (tindakan)
  - Jelaskan DBEngine Versi (tindakan)
  - Struktur:

- DBEngineVersi (struktur)
- EngineDefaults (struktur)
- PendingMaintenanceAction (struktur)
- ResourcePendingMaintenanceActions (struktur)
- UpgradeTarget (struktur)
- Tag (Struktur)
- Datatype Neptune Umum
  - AvailabilityZone (struktur)
  - DBSecurityGroupMembership (struktur)
  - DomainMembership (struktur)
  - DoubleRange (struktur)
  - Titik akhir (struktur)
  - Filter (struktur)
  - Kisaran (struktur)
  - ServerlessV2 (struktur) ScalingConfiguration
  - ServerlessV2 (struktur) ScalingConfigurationInfo
  - Zona waktu (struktur)
  - VpcSecurityGroupMembership (struktur)
- Pengecualian Neptunus Khusus untuk Individu APIs
  - AuthorizationAlreadyExistsFault (struktur)
  - AuthorizationNotFoundFault (struktur)
  - AuthorizationQuotaExceededFault (struktur)
  - CertificateNotFoundFault (struktur)
  - DBClusterAlreadyExistsFault (struktur)
  - DBClusterNotFoundFault (struktur)
  - DBClusterParameterGroupNotFoundFault (struktur)
  - DBClusterQuotaExceededFault (struktur)
  - DBClusterRoleAlreadyExistsFault (struktur)
  - DBClusterRoleNotFoundFault (struktur)
  - DBClusterRoleQuotaExceededFault (struktur)

- DBClusterSnapshotAlreadyExistsFault (struktur)
- DBClusterSnapshotNotFoundFault (struktur)
- DBInstanceAlreadyExistsFault (struktur)
- DBInstanceNotFoundFault (struktur)
- DBLogFileNotFoundFault (struktur)
- DBParameterGroupAlreadyExistsFault (struktur)
- DBParameterGroupNotFoundFault (struktur)
- DBParameterGroupQuotaExceededFault (struktur)
- DBSecurityGroupAlreadyExistsFault (struktur)
- DBSecurityGroupNotFoundFault (struktur)
- DBSecurityGroupNotSupportedFault (struktur)
- DBSecurityGroupQuotaExceededFault (struktur)
- DBSnapshotAlreadyExistsFault (struktur)
- DBSnapshotNotFoundFault (struktur)
- DBSubnetGroupAlreadyExistsFault (struktur)
- DBSubnetGroupDoesNotCoverEnoughAZs (struktur)
- DBSubnetGroupNotAllowedFault (struktur)
- DBSubnetGroupNotFoundFault (struktur)
- DBSubnetGroupQuotaExceededFault (struktur)
- DBSubnetQuotaExceededFault (struktur)
- DBUpgradeDependencyFailureFault (struktur)
- DomainNotFoundFault (struktur)
- EventSubscriptionQuotaExceededFault (struktur)
- GlobalClusterAlreadyExistsFault (struktur)
- GlobalClusterNotFoundFault (struktur)
- GlobalClusterQuotaExceededFault (struktur)
- InstanceQuotaExceededFault (struktur)
- Tidak cukup DBCluster CapacityFault (struktur)
- Tidak cukup DBInstance CapacityFault (struktur)
- InsufficientStorageClusterCapacityFault (struktur)

- Tidak valid DBCluster EndpointStateFault (struktur)
- Tidak valid DBCluster SnapshotStateFault (struktur)
- Tidak valid DBCluster StateFault (struktur)
- Tidak valid DBInstance StateFault (struktur)
- Tidak valid DBParameter GroupStateFault (struktur)
- Tidak valid DBSecurity GroupStateFault (struktur)
- Tidak valid DBSnapshot StateFault (struktur)
- Tidak valid DBSubnet GroupFault (struktur)
- Tidak valid DBSubnet GroupStateFault (struktur)
- Tidak valid DBSubnet StateFault (struktur)
- InvalidEventSubscriptionStateFault (struktur)
- InvalidGlobalClusterStateFault (struktur)
- InvalidOptionGroupStateFault (struktur)
- InvalidRestoreFault (struktur)
- InvalidSubnet (struktur)
- Tidak valid VPCNetwork StateFault (struktur)
- KMSKeyNotAccessibleFault (struktur)
- OptionGroupNotFoundFault (struktur)
- PointInTimeRestoreNotEnabledFault (struktur)
- ProvisionedIopsNotAvailableInAZFault (struktur)
- ResourceNotFoundFault (struktur)
- SNSInvalidTopicFault (struktur)
- SNSNoAuthorizationFault (struktur)
- SNSTopicArnNotFoundFault (struktur)
- SharedSnapshotQuotaExceededFault (struktur)
- SnapshotQuotaExceededFault (struktur)
- SourceNotFoundFault (struktur)
- StorageQuotaExceededFault (struktur)
- StorageTypeNotSupportedFault (struktur)
- SubnetAlreadyInUse (struktur)

- SubscriptionAlreadyExistFault (struktur)
- SubscriptionCategoryNotFoundFault (struktur)
- SubscriptionNotFoundFault (struktur)

API Klaster DB Neptune

Tindakan:

- Buat DBCluster (tindakan)
- Hapus DBCluster (tindakan)
- Memodifikasi DBCluster (tindakan)
- Mulai DBCluster (tindakan)
- Berhenti DBCluster (tindakan)
- AddRoleToDBCluster (tindakan)
- RemoveRoleFromDBCluster (tindakan)
- Failover DBCluster (tindakan)
- PromoteReadReplicaDBCluster (tindakan)
- Jelaskan DBClusters (tindakan)

Struktur:

- DBCluster (struktur)
- DBClusterAnggota (struktur)
- DBClusterPeran (struktur)
- CloudwatchLogsExportConfiguration (struktur)
- PendingCloudwatchLogsExports (struktur)
- ClusterPendingModifiedValues (struktur)

Buat DBCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `create-db-cluster`

Menciptakan klaster DB Amazon Neptune baru.

Anda dapat menggunakan parameter `ReplicationSourceIdentifier` untuk membuat klaster DB sebagai Replika Baca klaster DB lain atau instans DB Amazon Neptune.

Perhatikan bahwa ketika Anda membuat sebuah klaster baru menggunakan `CreateDBCluster` secara langsung, perlindungan penghapusan dinonaktifkan secara default (ketika Anda membuat klaster produksi baru di konsol, perlindungan penghapusan diaktifkan secara default). Anda hanya dapat menghapus klaster DB jika bidang `DeletionProtection` diatur ke `false`.

Permintaan

- AvailabilityZones(dalam CLI:`--availability-zones`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar EC2 Availability Zones yang instance di cluster DB dapat dibuat di.
- BackupRetentionPeriod(dalam CLI:`--backup-retention-period`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah hari penyimpanan cadangan otomatis. Anda harus menentukan nilai minimum 1.

  Default: 1

  Batasan:
  - Harus berupa nilai dari 1 hingga 35
- CopyTagsToSnapshot(dalam CLI:`--copy-tags-to-snapshot`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke*`true`*, tag akan disalin ke snapshot apa pun dari cluster DB yang dibuat.
- DatabaseName(dalam CLI:`--database-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama untuk basis data Anda hingga 64 karakter alfanumerik. Jika Anda tidak memberikan nama, Amazon Neptune tidak akan membuat basis data di klaster DB yang sedang Anda buat.
- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi klaster DB. Parameter ini disimpan sebagai string huruf kecil.

  Batas:

- Harus berisi 1 sampai 63 huruf, angka, atau tanda hubung.
- Karakter pertama harus berupa huruf.
- Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.

Contoh: `my-cluster1`

- DBClusterParameterGroupName(dalam CLI:`--db-cluster-parameter-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter klaster yang akan dikaitkan dengan klaster DB ini. Jika argumen ini dihilangkan, defaultnya digunakan.

  Batasan:

  - Jika disediakan, harus cocok dengan nama yang sudah ada DBClusterParameterGroup.
- DBSubnetGroupName(dalam CLI:`--db-subnet-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Grup subnet DB yang akan dikaitkan dengan klaster DB ini.

  Kendala: Harus cocok dengan nama Grup yang ada. DBSubnet Tidak harus default.

  Contoh: `mySubnetgroup`
- DeletionProtection(dalam CLI:`--deletion-protection`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang menunjukkan apakah klaster DB memiliki perlindungan penghapusan yang diaktifkan. Basis data tidak dapat dihapus saat perlindungan penghapusan diaktifkan. Perlindungan penghapusan diaktifkan secara default.
- EnableCloudwatchLogsExports(dalam CLI:`--enable-cloudwatch-logs-exports`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang harus diekspor oleh cluster DB ini ke CloudWatch Log. Jenis log yang valid adalah: `audit` (untuk mempublikasikan log audit) dan `slowquery` (untuk menerbitkan log kueri lambat). Lihat Menerbitkan log Neptunus ke log Amazon. CloudWatch
- EnableIAMDatabaseAuthentication(dalam CLI:`--enable-iam-database-authentication`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika diatur ke `true`, mengaktifkan autentikasi Amazon Identity and Access Management (IAM) untuk keseluruhan klaster DB (ini tidak dapat diatur pada tingkat instans).

Default: `false`.

- Engine(dalam CLI:`--engine`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama mesin basis data yang akan digunakan untuk klaster DB ini.

  Nilai Valid: `neptune`

- EngineVersion(dalam CLI:`--engine-version`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nomor versi mesin basis data yang akan digunakan untuk klaster DB baru.

  Contoh: `1.2.1.0`

- GlobalClusterIdentifier(dalam CLI:`--global-cluster-identifier`) — a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  ID database global Neptunus yang harus ditambahkan kluster DB baru ini.

- KmsKeyId(dalam CLI:`--kms-key-id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi kunci Amazon KMS untuk klaster DB terenkripsi.

  Pengidentifikasi kunci KMS adalah Amazon Resource Name (ARN) untuk kunci enkripsi KMS. Jika Anda membuat klaster DB dengan akun Amazon yang sama yang memiliki kunci enkripsi KMS yang digunakan untuk mengenkripsi klaster DB baru, maka Anda dapat menggunakan alias kunci KMS alih-alih ARN untuk kunci enkripsi KMS.

  Jika kunci enkripsi tidak ditentukan dalam `KmsKeyId`:

  - Jika `ReplicationSourceIdentifier` mengidentifikasi sumber terenkripsi, Amazon Neptune akan menggunakan kunci enkripsi yang digunakan untuk mengenkripsi sumbernya. Jika tidak, Amazon Neptune akan menggunakan kunci enkripsi default Anda.
  - Jika parameter `StorageEncrypted` benar dan `ReplicationSourceIdentifier` tidak ditentukan, maka Amazon Neptune akan menggunakan kunci enkripsi default Anda.

  Amazon KMS menciptakan kunci enkripsi default untuk akun Amazon Anda. Akun Amazon Anda memiliki kunci enkripsi default yang berbeda untuk setiap Wilayah Amazon.

Jika Anda membuat Replika Baca klaster DB terenkripsi di Wilayah Amazon lain, Anda harus mengatur `KmsKeyId` ke ID kunci KMS yang berlaku di Wilayah Amazon tujuan. Kunci ini digunakan untuk mengenkripsi Replika Baca di Wilayah Amazon tersebut.

- Port(dalam CLI:`--port`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Nomor port tempat instans dalam klaster DB menerima koneksi.

  Default: `8182`

- PreferredBackupWindow(dalam CLI:`--preferred-backup-window`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Rentang waktu harian selama pencadangan otomatis dibuat jika pencadangan otomatis diaktifkan menggunakan parameter `BackupRetentionPeriod`.

  Default-nya adalah jendela 30 menit yang dipilih secara acak dari blok waktu 8 jam untuk setiap Wilayah Amazon. Untuk melihat blok waktu yang tersedia, lihat Jendela Pemeliharaan Neptunus di Panduan Pengguna Amazon Neptunus.

  Batasan:

  - Harus dalam format `hh24:mi-hh24:mi`.
  - Harus dalam Waktu Universal Terkoordinasi (UTC).
  - Tidak boleh bertentangan dengan jendela pemeliharaan yang diinginkan.
  - Harus setidaknya 30 menit.

- PreferredMaintenanceWindow(dalam CLI:`--preferred-maintenance-window`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Rentang waktu mingguan selama pemeliharaan sistem dapat terjadi, dalam Waktu Universal Terkoordinasi (UTC).

  Format: `ddd:hh24:mi-ddd:hh24:mi`

  Default adalah jendela 30 menit yang dipilih secara acak dari blok waktu 8 jam per Wilayah Amazon, yang dilakukan pada sembarang hari dalam seminggu. Untuk melihat blok waktu yang tersedia, lihat Jendela Pemeliharaan Neptunus di Panduan Pengguna Amazon Neptunus.

  Hari yang valid: Sen, Sel, Rab, Kam, Jum, Sab, Min.

Kendala: Minimum 30 menit jendela.

- PreSignedUrl(dalam CLI:`--pre-signed-url`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Parameter ini saat ini tidak didukung.

- ReplicationSourceIdentifier(dalam CLI:`--replication-source-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) dari instans DB sumber atau klaster DB jika klaster DB ini dibuat sebagai sebuah Replika Baca.

- ServerlessV2ScalingConfiguration(dalam CLI:`--serverless-v2-scaling-configuration`) — Sebuah ServerlessV2 ScalingConfiguration objek.

  Berisi konfigurasi penskalaan cluster DB Neptunus Tanpa Server.

  Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- StorageEncrypted(dalam CLI:`--storage-encrypted`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB dienkripsi.

- StorageType(dalam CLI:`--storage-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan untuk cluster DB baru.

  Nilai yang Valid:

  - **`standard`**— (default) Mengkonfigurasi penyimpanan database hemat biaya untuk aplikasi dengan penggunaan sedang hingga kecil. I/O Saat disetel k`estandard`, jenis penyimpanan tidak dikembalikan dalam respons.
  - **`iopt1`**— Memungkinkan penyimpanan I/O-Optimized yang dirancang untuk memenuhi kebutuhan throughput. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    Penyimpanan Neptunus I/O-Optimized hanya tersedia dimulai dengan rilis engine 1.3.0.0.

- Tags(dalam CLI:`--tags`) — Sebuah array objek. Tag

Tanda yang akan ditetapkan ke klaster DB baru.

- VpcSecurityGroupIds(dalam CLI:`--vpc-security-group-ids`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar grup keamanan EC2 VPC untuk dikaitkan dengan cluster DB ini.

Respons

Berisi detail dari klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam filethe section called “Jelaskan DBClusters”.

- AllocatedStorage— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  `AllocatedStorage` selalu mengembalikan 1, karena ukuran penyimpanan klaster DB Neptune tidak tetap, tetapi justru secara otomatis menyesuaikan sesuai kebutuhan.
- AssociatedRoles – Susunan objek DBClusterPeran.

  Menyediakan daftar Amazon Identity and Access Management (IAM) role yang terkait dengan klaster DB. IAM role yang terkait dengan klaster DB memberikan izin pada klaster DB untuk mengakses layanan Amazon lain atas nama Anda.
- AutomaticRestartTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Waktu di mana klaster DB akan secara otomatis di-restart.
- AvailabilityZones— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance di cluster DB dapat dibuat.
- BacktrackConsumedChangeRecords— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.
- BacktrackWindow— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.
- BackupRetentionPeriod— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.

- Capacity— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Tidak didukung oleh Neptune.
- CloneGroupId— String, tipe: `string` (string yang dikodekan UTF-8).

  Mengidentifikasi grup klon yang terkait dengan klaster DB.
- ClusterCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).
- CopyTagsToSnapshot— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke*`true`*, tag akan disalin ke snapshot apa pun dari cluster DB yang dibuat.
- CrossAccountClone— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika diatur ke `true`, klaster DB dapat dikloning di seluruh akun.
- DatabaseName— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama basis data awal klaster DB ini yang disediakan pada waktu membuat, jika salah satu ditentukan ketika klaster DB dibuat. Nama yang sama ini dikembalikan demi kluster DB.
- DBClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk klaster DB.
- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi klaster DB yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi klaster DB.
- DBClusterMembers – Susunan objek DBClusterAnggota.

  Menyediakan daftar instans yang membentuk klaster DB.
- DBClusterParameterGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama grup parameter klaster DB untuk klaster DB ini.
- DbClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi Wilayah Amazon unik yang tetap untuk klaster. Pengidentifikasi ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk cluster DB diakses.
- DBSubnetGroup— String, tipe: `string` (string yang dikodekan UTF-8).

Menentukan informasi pada grup subnet yang terkait dengan klaster DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan.

- EarliestBacktrackTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tidak didukung oleh Neptune.

- EarliestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu paling awal dimana database dapat dipulihkan dengan point-in-time restore.

- EnabledCloudwatchLogsExports— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang cluster DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log. Jenis log yang valid adalah: `audit` (untuk mempublikasikan log audit ke CloudWatch) dan slowquery (untuk mempublikasikan log kueri lambat ke). CloudWatch Lihat Menerbitkan log Neptunus ke log Amazon. CloudWatch

- Endpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan titik akhir koneksi untuk instans primer klaster DB.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk klaster DB ini.

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.

- HostedZoneId— String, tipe: `string` (string yang dikodekan UTF-8).

Menentukan ID yang ditetapkan Amazon Route 53 saat Anda membuat zona yang di-hosting.

- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.

- IOOptimizedNextAllowedModificationTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Lain kali Anda dapat memodifikasi cluster DB untuk menggunakan jenis `iopt1` penyimpanan.

- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah BETUL, pengidentifikasi kunci Amazon KMS untuk klaster DB terenkripsi.

- LatestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.

- MultiAZ— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB memiliki instans di beberapa Availability Zone.

- PendingModifiedValues — Sebuah objek ClusterPendingModifiedValues.

  Tipe data ini digunakan sebagai elemen respons dalam `ModifyDBCluster` operasi dan berisi perubahan yang akan diterapkan selama jendela pemeliharaan berikutnya.

- PercentProgress— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan kemajuan operasi sebagai persentase.

- Port— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan oleh mesin basis data.

- PreferredBackupWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.

- PreferredMaintenanceWindow— String, tipe: `string` (string yang dikodekan UTF-8).

Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).

- ReaderEndpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Reader Endpoint untuk klaster DB. Reader Endpoint untuk koneksi menyeimbangkan beban klaster DB di seluruh Replika Baca yang tersedia dalam klaster DB. Saat klien meminta koneksi baru ke Reader Endpoint, Neptune mendistribusikan permintaan koneksi di antara Replika Neptune di klaster DB. Fungsionalitas ini dapat membantu menyeimbangkan beban kerja baca Anda di beberapa Replika Baca di klaster DB Anda.

  Jika terjadi failover, dan Replika Baca yang terhubung dengan Anda dipromosikan menjadi instans primer, koneksi Anda dijatuhkan. Untuk terus mengirimkan beban kerja baca Anda ke Replika Baca lainnya di klaster, Anda kemudian dapat menyambungkan kembali ke Reader Endpoint.

- ReadReplicaIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi Replika Baca terkait dengan klaster DB ini.

- ReplicationSourceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ReplicationType— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ServerlessV2ScalingConfiguration — Sebuah objek ServerlessV2 ScalingConfigurationInfo.

  Menampilkan konfigurasi penskalaan untuk cluster DB Neptunus Tanpa Server.

  Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari klaster DB ini.

- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB dienkripsi.

- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang digunakan oleh cluster DB.

Nilai yang Valid:

- **`standard`**— (default) Menyediakan penyimpanan database hemat biaya untuk aplikasi dengan penggunaan sedang hingga kecil I/O .
- **`iopt1`**— Memungkinkan penyimpanan I/O-Optimized yang dirancang untuk memenuhi kebutuhan throughput. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

  Penyimpanan Neptunus I/O-Optimized hanya tersedia dimulai dengan rilis engine 1.3.0.0.

- VpcSecurityGroups – Susunan objek VpcSecurityGroupMembership.

  Memberikan daftar grup keamanan VPC yang memiliki klaster DB.

Kesalahan

- DBClusterAlreadyExistsFault
- InsufficientStorageClusterCapacityFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- Tidak valid VPCNetwork StateFault
- Tidak valid DBCluster StateFault
- Tidak valid DBSubnet GroupStateFault
- InvalidSubnet
- Tidak valid DBInstance StateFault
- DBClusterParameterGroupNotFoundFault
- KMSKeyNotAccessibleFault
- DBClusterNotFoundFault
- DBInstanceNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

Hapus DBCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `delete-db-cluster`

DBCluster Tindakan Hapus menghapus klaster DB yang telah disediakan sebelumnya. Ketika Anda menghapus sebuah klaster DB, semua backup otomatis untuk kluster DB tersebut dihapus dan tidak dapat dipulihkan. Snapshot klaster DB manual dari klaster DB yang ditentukan tidak dihapus.

Ingat bahwa klaster DB tidak dapat dihapus jika perlindungan penghapusan diaktifkan. Untuk menghapusnya, Anda mesti terlebih dahulu menetapkan bidang `DeletionProtection` ke `False`.

Permintaan

- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi klaster DB untuk kluster DB yang akan dihapus. Parameter ini tidak peka huruf besar-kecil.

  Batasan:

  - Harus cocok dengan DBCluster Identifier yang ada.
- FinalDBSnapshotIdentifier(dalam CLI:`--final-db-snapshot-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi snapshot klaster DB dari snapshot klaster DB baru yang dibuat saat `SkipFinalSnapshot` diatur ke `false`.

  > **Note**
  >
  > Menentukan parameter ini dan juga mengatur parameter `SkipFinalShapshot` ke true menghasilkan kesalahan.

  Batasan:

  - Harus berisi 1 sampai 255 huruf, angka, atau tanda hubung.
  - Karakter pertamanya harus berupa huruf
  - Tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung yang berurutan

- SkipFinalSnapshot(dalam CLI:`--skip-final-snapshot`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah snapshot klaster DB akhir dibuat sebelum klaster DB dihapus. Jika `true` ditentukan, tidak ada snapshot klaster DB yang dibuat. Jika `false` ditentukan, snapshot klaster DB dibuat sebelum klaster DB dihapus.

  > **Note**
  >
  > Anda harus menentukan parameter `FinalDBSnapshotIdentifier` jika `SkipFinalSnapshot` adalah `false`.

  Default: `false`

Respons

Berisi detail dari klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam filethe section called “Jelaskan DBClusters”.

- AllocatedStorage— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  `AllocatedStorage` selalu mengembalikan 1, karena ukuran penyimpanan klaster DB Neptune tidak tetap, tetapi justru secara otomatis menyesuaikan sesuai kebutuhan.
- AssociatedRoles – Susunan objek DBClusterPeran.

  Menyediakan daftar Amazon Identity and Access Management (IAM) role yang terkait dengan klaster DB. IAM role yang terkait dengan klaster DB memberikan izin pada klaster DB untuk mengakses layanan Amazon lain atas nama Anda.
- AutomaticRestartTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Waktu di mana klaster DB akan secara otomatis di-restart.
- AvailabilityZones— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance di cluster DB dapat dibuat.
- BacktrackConsumedChangeRecords— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

Tidak didukung oleh Neptune.

- BacktrackWindow— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- BackupRetentionPeriod— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.

- Capacity— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- CloneGroupId— String, tipe: `string` (string yang dikodekan UTF-8).

  Mengidentifikasi grup klon yang terkait dengan klaster DB.

- ClusterCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).

- CopyTagsToSnapshot— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke*`true`*, tag akan disalin ke snapshot apa pun dari cluster DB yang dibuat.

- CrossAccountClone— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika diatur ke `true`, klaster DB dapat dikloning di seluruh akun.

- DatabaseName— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama basis data awal klaster DB ini yang disediakan pada waktu membuat, jika salah satu ditentukan ketika klaster DB dibuat. Nama yang sama ini dikembalikan demi kluster DB.

- DBClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk klaster DB.

- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi klaster DB yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi klaster DB.

- DBClusterMembers – Susunan objek DBClusterAnggota.

  Menyediakan daftar instans yang membentuk klaster DB.

- DBClusterParameterGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama grup parameter klaster DB untuk klaster DB ini.

- DbClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi Wilayah Amazon unik yang tetap untuk klaster. Pengidentifikasi ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk cluster DB diakses.

- DBSubnetGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan informasi pada grup subnet yang terkait dengan klaster DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan.

- EarliestBacktrackTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tidak didukung oleh Neptune.

- EarliestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu paling awal dimana database dapat dipulihkan dengan point-in-time restore.

- EnabledCloudwatchLogsExports— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang cluster DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log. Jenis log yang valid adalah: `audit` (untuk mempublikasikan log audit ke CloudWatch) dan slowquery (untuk mempublikasikan log kueri lambat ke). CloudWatch Lihat Menerbitkan log Neptunus ke log Amazon. CloudWatch

- Endpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan titik akhir koneksi untuk instans primer klaster DB.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk klaster DB ini.

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.
- HostedZoneId— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan ID yang ditetapkan Amazon Route 53 saat Anda membuat zona yang di-hosting.
- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.
- IOOptimizedNextAllowedModificationTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Lain kali Anda dapat memodifikasi cluster DB untuk menggunakan jenis `iopt1` penyimpanan.
- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah BETUL, pengidentifikasi kunci Amazon KMS untuk klaster DB terenkripsi.
- LatestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.
- MultiAZ— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB memiliki instans di beberapa Availability Zone.
- PendingModifiedValues — Sebuah objek ClusterPendingModifiedValues.

  Tipe data ini digunakan sebagai elemen respons dalam `ModifyDBCluster` operasi dan berisi perubahan yang akan diterapkan selama jendela pemeliharaan berikutnya.
- PercentProgress— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan kemajuan operasi sebagai persentase.
- Port— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan oleh mesin basis data.

- PreferredBackupWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.

- PreferredMaintenanceWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).

- ReaderEndpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Reader Endpoint untuk klaster DB. Reader Endpoint untuk koneksi menyeimbangkan beban klaster DB di seluruh Replika Baca yang tersedia dalam klaster DB. Saat klien meminta koneksi baru ke Reader Endpoint, Neptune mendistribusikan permintaan koneksi di antara Replika Neptune di klaster DB. Fungsionalitas ini dapat membantu menyeimbangkan beban kerja baca Anda di beberapa Replika Baca di klaster DB Anda.

  Jika terjadi failover, dan Replika Baca yang terhubung dengan Anda dipromosikan menjadi instans primer, koneksi Anda dijatuhkan. Untuk terus mengirimkan beban kerja baca Anda ke Replika Baca lainnya di klaster, Anda kemudian dapat menyambungkan kembali ke Reader Endpoint.

- ReadReplicaIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi Replika Baca terkait dengan klaster DB ini.

- ReplicationSourceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ReplicationType— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ServerlessV2ScalingConfiguration — Sebuah objek ServerlessV2 ScalingConfigurationInfo.

  Menampilkan konfigurasi penskalaan untuk cluster DB Neptunus Tanpa Server.

  Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari klaster DB ini.

- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

Menentukan apakah klaster DB dienkripsi.

- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang digunakan oleh cluster DB.

  Nilai yang Valid:

  - **`standard`**— (default) Menyediakan penyimpanan database hemat biaya untuk aplikasi dengan penggunaan sedang hingga kecil I/O .
  - **`iopt1`**— Memungkinkan penyimpanan I/O-Optimized yang dirancang untuk memenuhi kebutuhan throughput. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    Penyimpanan Neptunus I/O-Optimized hanya tersedia dimulai dengan rilis engine 1.3.0.0.
- VpcSecurityGroups – Susunan objek VpcSecurityGroupMembership.

  Memberikan daftar grup keamanan VPC yang memiliki klaster DB.

Kesalahan

- DBClusterNotFoundFault
- Tidak valid DBCluster StateFault
- DBClusterSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault
- Tidak valid DBCluster SnapshotStateFault

Memodifikasi DBCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `modify-db-cluster`

Memodifikasi pengaturan untuk klaster DB. Anda dapat mengubah satu atau lebih parameter konfigurasi basis data dengan menentukan parameter ini dan nilai-nilai baru dalam permintaan.

Permintaan

- AllowMajorVersionUpgrade(dalam CLI:`--allow-major-version-upgrade`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

Nilai yang menunjukkan apakah peningkatan antara versi utama yang berbeda diperbolehkan.

Constraints: Anda harus menyetel allow-major-version-upgrade flag saat menyediakan `EngineVersion` parameter yang menggunakan versi mayor yang berbeda dari versi cluster DB saat ini.

- ApplyImmediately(dalam CLI:`--apply-immediately`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang menentukan apakah modifikasi dalam permintaan ini dan modifikasi yang tertunda diterapkan secara asinkron sesegera mungkin, terlepas dari pengaturan `PreferredMaintenanceWindow` untuk klaster DB. Jika parameter ini diatur ke `false`, perubahan ke klaster DB diterapkan selama jendela pemeliharaan berikutnya.

  Parameter `ApplyImmediately` hanya mempengaruhi nilai `NewDBClusterIdentifier`. Jika Anda mengatur nilai parameter `ApplyImmediately` ke false, maka perubahan ke nilai `NewDBClusterIdentifier` diterapkan selama jendela pemeliharaan berikutnya. Semua perubahan lainnya akan diterapkan dengan segera, terlepas dari nilai parameter `ApplyImmediately`.

  Default: `false`

- BackupRetentionPeriod(dalam CLI:`--backup-retention-period`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah hari penyimpanan cadangan otomatis. Anda harus menentukan nilai minimum 1.

  Default: 1

  Batasan:

  - Harus berupa nilai dari 1 hingga 35

- CloudwatchLogsExportConfiguration(dalam CLI:`--cloudwatch-logs-export-configuration`) — Sebuah CloudwatchLogsExportConfiguration objek.

  Pengaturan konfigurasi untuk jenis log yang akan diaktifkan untuk ekspor ke CloudWatch Log untuk cluster DB tertentu. Lihat Menggunakan CLI untuk mempublikasikan log audit Neptunus ke Log. CloudWatch

- CopyTagsToSnapshot(dalam CLI:`--copy-tags-to-snapshot`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

Jika disetel ke*true*, tag akan disalin ke snapshot apa pun dari cluster DB yang dibuat.

- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi klaster DB untuk klaster yang sedang dimodifikasi. Parameter ini tidak peka huruf besar kecil.

  Batasan:

  - Harus cocok dengan pengenal yang sudah ada DBCluster.
- DBClusterParameterGroupName(dalam CLI:`--db-cluster-parameter-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter klaster DB yang akan digunakan untuk klaster DB ini.
- DBInstanceParameterGroupName(dalam CLI:`--db-instance-parameter-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter DB untuk diterapkan ke semua instance cluster DB.

  > **Note**
  >
  > Saat Anda menerapkan grup parameter menggunakan`DBInstanceParameterGroupName`, perubahan parameter tidak diterapkan selama jendela pemeliharaan berikutnya tetapi diterapkan segera.

  Default: Pengaturan nama yang ada

  Batasan:

  - Grup parameter DB harus berada dalam keluarga grup parameter DB yang sama dengan versi cluster DB target.
  - `DBInstanceParameterGroupName`Parameter hanya valid dalam kombinasi dengan `AllowMajorVersionUpgrade` parameter.
- DeletionProtection(dalam CLI:`--deletion-protection`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

Nilai yang menunjukkan apakah klaster DB memiliki perlindungan penghapusan yang diaktifkan. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan. Secara default, perlindungan penghapusan dinonaktifkan.

- EnableIAMDatabaseAuthentication(dalam CLI:`--enable-iam-database-authentication`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  True untuk mengaktifkan pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data, dan sebaliknya false.

  Default: `false`

- EngineVersion(dalam CLI:`--engine-version`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nomor versi mesin basis data yang ingin Anda tingkatkan. Mengubah parameter ini menghasilkan pemadaman. Kecuali parameter `ApplyImmediately` diatur ke true, perubahan akan diterapkan selama jendela pemeliharaan berikutnya.

  Untuk daftar versi engine yang valid, lihat Rilis Mesin untuk Amazon Neptunus, atau hubungi. the section called “Jelaskan DBEngine Versi”

- NewDBClusterIdentifier(dalam CLI:`--new-db-cluster-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi klaster DB baru untuk klaster DB ketika mengubah nama klaster DB. Nilai ini disimpan sebagai string huruf kecil.

  Batasan:

  - Harus berisi 1 sampai 63 huruf, angka, atau tanda hubung
  - Karakter pertama harus berupa huruf
  - Tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung yang berurutan

  Contoh: `my-cluster2`

- Port(dalam CLI:`--port`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Angka port tempat klaster DB menerima koneksi.

  Kendala: Nilai harus `1150-65535`

Default: Port yang sama dengan klaster DB asli.

- PreferredBackupWindow(dalam CLI:`--preferred-backup-window`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Kisaran waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, menggunakan parameter `BackupRetentionPeriod`.

  Default-nya adalah jendela 30 menit yang dipilih secara acak dari blok waktu 8 jam untuk setiap Wilayah Amazon.

  Batasan:

  - Harus dalam format `hh24:mi-hh24:mi`.
  - Harus dalam Waktu Universal Terkoordinasi (UTC).
  - Tidak boleh bertentangan dengan jendela pemeliharaan yang diinginkan.
  - Harus setidaknya 30 menit.
- PreferredMaintenanceWindow(dalam CLI:`--preferred-maintenance-window`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Rentang waktu mingguan selama pemeliharaan sistem dapat terjadi, dalam Waktu Universal Terkoordinasi (UTC).

  Format: `ddd:hh24:mi-ddd:hh24:mi`

  Default adalah jendela 30 menit yang dipilih secara acak dari blok waktu 8 jam per Wilayah Amazon, yang dilakukan pada sembarang hari dalam seminggu.

  Hari yang valid: Sen, Sel, Rab, Kam, Jum, Sab, Min.

  Kendala: Minimum 30 menit jendela.
- ServerlessV2ScalingConfiguration(dalam CLI:`--serverless-v2-scaling-configuration`) — Sebuah ServerlessV2 ScalingConfiguration objek.

  Berisi konfigurasi penskalaan cluster DB Neptunus Tanpa Server.

  Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- StorageType(dalam CLI:`--storage-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan untuk mengasosiasikan dengan cluster DB.

  Nilai yang Valid:

  - **`standard`**— (default) Mengkonfigurasi penyimpanan database hemat biaya untuk aplikasi dengan penggunaan sedang hingga kecil. I/O
  - **`iopt1`**— Memungkinkan penyimpanan I/O-Optimized yang dirancang untuk memenuhi kebutuhan throughput. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    Penyimpanan Neptunus I/O-Optimized hanya tersedia dimulai dengan rilis engine 1.3.0.0.
- VpcSecurityGroupIds(dalam CLI:`--vpc-security-group-ids`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar grup keamanan VPC tempat klaster DB baru akan berada.

Respons

Berisi detail dari klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam filethe section called “Jelaskan DBClusters”.

- AllocatedStorage— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  `AllocatedStorage` selalu mengembalikan 1, karena ukuran penyimpanan klaster DB Neptune tidak tetap, tetapi justru secara otomatis menyesuaikan sesuai kebutuhan.
- AssociatedRoles – Susunan objek DBClusterPeran.

  Menyediakan daftar Amazon Identity and Access Management (IAM) role yang terkait dengan klaster DB. IAM role yang terkait dengan klaster DB memberikan izin pada klaster DB untuk mengakses layanan Amazon lain atas nama Anda.
- AutomaticRestartTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Waktu di mana klaster DB akan secara otomatis di-restart.
- AvailabilityZones— String, tipe: `string` (string yang dikodekan UTF-8).

Menyediakan daftar EC2 Availability Zones dimana instance di cluster DB dapat dibuat.

- BacktrackConsumedChangeRecords— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- BacktrackWindow— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- BackupRetentionPeriod— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.

- Capacity— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- CloneGroupId— String, tipe: `string` (string yang dikodekan UTF-8).

  Mengidentifikasi grup klon yang terkait dengan klaster DB.

- ClusterCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).

- CopyTagsToSnapshot— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke*`true`*, tag akan disalin ke snapshot apa pun dari cluster DB yang dibuat.

- CrossAccountClone— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika diatur ke `true`, klaster DB dapat dikloning di seluruh akun.

- DatabaseName— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama basis data awal klaster DB ini yang disediakan pada waktu membuat, jika salah satu ditentukan ketika klaster DB dibuat. Nama yang sama ini dikembalikan demi kluster DB.

- DBClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk klaster DB.

- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

Berisi pengidentifikasi klaster DB yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi klaster DB.

- DBClusterMembers – Susunan objek DBClusterAnggota.

  Menyediakan daftar instans yang membentuk klaster DB.

- DBClusterParameterGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama grup parameter klaster DB untuk klaster DB ini.

- DbClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi Wilayah Amazon unik yang tetap untuk klaster. Pengidentifikasi ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk cluster DB diakses.

- DBSubnetGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan informasi pada grup subnet yang terkait dengan klaster DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan.

- EarliestBacktrackTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tidak didukung oleh Neptune.

- EarliestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu paling awal dimana database dapat dipulihkan dengan point-in-time restore.

- EnabledCloudwatchLogsExports— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang cluster DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log. Jenis log yang valid adalah: `audit` (untuk mempublikasikan log audit ke CloudWatch) dan slowquery (untuk mempublikasikan log kueri lambat ke). CloudWatch Lihat Menerbitkan log Neptunus ke log Amazon. CloudWatch

- Endpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan titik akhir koneksi untuk instans primer klaster DB.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk klaster DB ini.

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.

- HostedZoneId— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan ID yang ditetapkan Amazon Route 53 saat Anda membuat zona yang di-hosting.

- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.

- IOOptimizedNextAllowedModificationTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Lain kali Anda dapat memodifikasi cluster DB untuk menggunakan jenis `iopt1` penyimpanan.

- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah BETUL, pengidentifikasi kunci Amazon KMS untuk klaster DB terenkripsi.

- LatestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.

- MultiAZ— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB memiliki instans di beberapa Availability Zone.

- PendingModifiedValues — Sebuah objek ClusterPendingModifiedValues.

  Tipe data ini digunakan sebagai elemen respons dalam `ModifyDBCluster` operasi dan berisi perubahan yang akan diterapkan selama jendela pemeliharaan berikutnya.

- PercentProgress— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan kemajuan operasi sebagai persentase.

- Port— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan oleh mesin basis data.

- PreferredBackupWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.

- PreferredMaintenanceWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).

- ReaderEndpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Reader Endpoint untuk klaster DB. Reader Endpoint untuk koneksi menyeimbangkan beban klaster DB di seluruh Replika Baca yang tersedia dalam klaster DB. Saat klien meminta koneksi baru ke Reader Endpoint, Neptune mendistribusikan permintaan koneksi di antara Replika Neptune di klaster DB. Fungsionalitas ini dapat membantu menyeimbangkan beban kerja baca Anda di beberapa Replika Baca di klaster DB Anda.

  Jika terjadi failover, dan Replika Baca yang terhubung dengan Anda dipromosikan menjadi instans primer, koneksi Anda dijatuhkan. Untuk terus mengirimkan beban kerja baca Anda ke Replika Baca lainnya di klaster, Anda kemudian dapat menyambungkan kembali ke Reader Endpoint.

- ReadReplicaIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi Replika Baca terkait dengan klaster DB ini.

- ReplicationSourceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ReplicationType— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ServerlessV2ScalingConfiguration — Sebuah objek ServerlessV2 ScalingConfigurationInfo.

  Menampilkan konfigurasi penskalaan untuk cluster DB Neptunus Tanpa Server.

Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari klaster DB ini.

- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB dienkripsi.

- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang digunakan oleh cluster DB.

  Nilai yang Valid:

  - **`standard`**— (default) Menyediakan penyimpanan database hemat biaya untuk aplikasi dengan penggunaan sedang hingga kecil I/O .
  - **`iopt1`**— Memungkinkan penyimpanan I/O-Optimized yang dirancang untuk memenuhi kebutuhan throughput. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    Penyimpanan Neptunus I/O-Optimized hanya tersedia dimulai dengan rilis engine 1.3.0.0.

- VpcSecurityGroups – Susunan objek VpcSecurityGroupMembership.

  Memberikan daftar grup keamanan VPC yang memiliki klaster DB.

Kesalahan

- DBClusterNotFoundFault
- Tidak valid DBCluster StateFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- Tidak valid VPCNetwork StateFault
- Tidak valid DBSubnet GroupStateFault
- InvalidSubnet
- DBClusterParameterGroupNotFoundFault
- Tidak valid DBSecurity GroupStateFault

- Tidak valid DBInstance StateFault
- DBClusterAlreadyExistsFault
- StorageTypeNotSupportedFault

Mulai DBCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `start-db-cluster`

Memulai cluster Amazon Neptunus DB yang dihentikan menggunakan konsol Amazon, perintah Amazon stop-db-cluster CLI, atau Stop API. DBCluster

Permintaan

- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi klaster DB dari klaster DB Neptune akan dimulai. Parameter ini disimpan sebagai string huruf kecil.

Respons

Berisi detail dari klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam filethe section called “Jelaskan DBClusters”.

- AllocatedStorage— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  `AllocatedStorage` selalu mengembalikan 1, karena ukuran penyimpanan klaster DB Neptune tidak tetap, tetapi justru secara otomatis menyesuaikan sesuai kebutuhan.
- AssociatedRoles – Susunan objek DBClusterPeran.

  Menyediakan daftar Amazon Identity and Access Management (IAM) role yang terkait dengan klaster DB. IAM role yang terkait dengan klaster DB memberikan izin pada klaster DB untuk mengakses layanan Amazon lain atas nama Anda.
- AutomaticRestartTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Waktu di mana klaster DB akan secara otomatis di-restart.

- AvailabilityZones— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance di cluster DB dapat dibuat.

- BacktrackConsumedChangeRecords— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- BacktrackWindow— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- BackupRetentionPeriod— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.

- Capacity— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- CloneGroupId— String, tipe: `string` (string yang dikodekan UTF-8).

  Mengidentifikasi grup klon yang terkait dengan klaster DB.

- ClusterCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).

- CopyTagsToSnapshot— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke*`true`*, tag akan disalin ke snapshot apa pun dari cluster DB yang dibuat.

- CrossAccountClone— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika diatur ke `true`, klaster DB dapat dikloning di seluruh akun.

- DatabaseName— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama basis data awal klaster DB ini yang disediakan pada waktu membuat, jika salah satu ditentukan ketika klaster DB dibuat. Nama yang sama ini dikembalikan demi kluster DB.

- DBClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk klaster DB.

- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

Berisi pengidentifikasi klaster DB yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi klaster DB.

- DBClusterMembers – Susunan objek DBClusterAnggota.

  Menyediakan daftar instans yang membentuk klaster DB.

- DBClusterParameterGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama grup parameter klaster DB untuk klaster DB ini.

- DbClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi Wilayah Amazon unik yang tetap untuk klaster. Pengidentifikasi ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk cluster DB diakses.

- DBSubnetGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan informasi pada grup subnet yang terkait dengan klaster DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan.

- EarliestBacktrackTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tidak didukung oleh Neptune.

- EarliestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu paling awal dimana database dapat dipulihkan dengan point-in-time restore.

- EnabledCloudwatchLogsExports— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang cluster DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log. Jenis log yang valid adalah: `audit` (untuk mempublikasikan log audit ke CloudWatch) dan slowquery (untuk mempublikasikan log kueri lambat ke). CloudWatch Lihat Menerbitkan log Neptunus ke log Amazon. CloudWatch

- Endpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan titik akhir koneksi untuk instans primer klaster DB.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk klaster DB ini.

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.

- HostedZoneId— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan ID yang ditetapkan Amazon Route 53 saat Anda membuat zona yang di-hosting.

- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.

- IOOptimizedNextAllowedModificationTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Lain kali Anda dapat memodifikasi cluster DB untuk menggunakan jenis `iopt1` penyimpanan.

- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah BETUL, pengidentifikasi kunci Amazon KMS untuk klaster DB terenkripsi.

- LatestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.

- MultiAZ— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB memiliki instans di beberapa Availability Zone.

- PendingModifiedValues — Sebuah objek ClusterPendingModifiedValues.

  Tipe data ini digunakan sebagai elemen respons dalam `ModifyDBCluster` operasi dan berisi perubahan yang akan diterapkan selama jendela pemeliharaan berikutnya.

- PercentProgress— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan kemajuan operasi sebagai persentase.

- Port— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan oleh mesin basis data.

- PreferredBackupWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.

- PreferredMaintenanceWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).

- ReaderEndpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Reader Endpoint untuk klaster DB. Reader Endpoint untuk koneksi menyeimbangkan beban klaster DB di seluruh Replika Baca yang tersedia dalam klaster DB. Saat klien meminta koneksi baru ke Reader Endpoint, Neptune mendistribusikan permintaan koneksi di antara Replika Neptune di klaster DB. Fungsionalitas ini dapat membantu menyeimbangkan beban kerja baca Anda di beberapa Replika Baca di klaster DB Anda.

  Jika terjadi failover, dan Replika Baca yang terhubung dengan Anda dipromosikan menjadi instans primer, koneksi Anda dijatuhkan. Untuk terus mengirimkan beban kerja baca Anda ke Replika Baca lainnya di klaster, Anda kemudian dapat menyambungkan kembali ke Reader Endpoint.

- ReadReplicaIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi Replika Baca terkait dengan klaster DB ini.

- ReplicationSourceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ReplicationType— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ServerlessV2ScalingConfiguration — Sebuah objek ServerlessV2 ScalingConfigurationInfo.

  Menampilkan konfigurasi penskalaan untuk cluster DB Neptunus Tanpa Server.

Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari klaster DB ini.

- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB dienkripsi.

- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang digunakan oleh cluster DB.

  Nilai yang Valid:

  - **`standard`**— (default) Menyediakan penyimpanan database hemat biaya untuk aplikasi dengan penggunaan sedang hingga kecil I/O .
  - **`iopt1`**— Memungkinkan penyimpanan I/O-Optimized yang dirancang untuk memenuhi kebutuhan throughput. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    Penyimpanan Neptunus I/O-Optimized hanya tersedia dimulai dengan rilis engine 1.3.0.0.

- VpcSecurityGroups – Susunan objek VpcSecurityGroupMembership.

  Memberikan daftar grup keamanan VPC yang memiliki klaster DB.

Kesalahan

- DBClusterNotFoundFault
- Tidak valid DBCluster StateFault
- Tidak valid DBInstance StateFault

Berhenti DBCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `stop-db-cluster`

Menghentikan klaster DB Amazon Neptune. Ketika Anda menghentikan klaster DB, Neptune mempertahankan metadata klaster DB, termasuk titik akhir dan grup parameter DB.

Neptunus juga menyimpan log transaksi sehingga Anda dapat melakukan point-in-time pemulihan jika perlu.

Permintaan

- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi klaster DB dari klaster DB Neptune akan dihentikan. Parameter ini disimpan sebagai string huruf kecil.

Respons

Berisi detail dari klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam filethe section called “Jelaskan DBClusters”.

- AllocatedStorage— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  `AllocatedStorage` selalu mengembalikan 1, karena ukuran penyimpanan klaster DB Neptune tidak tetap, tetapi justru secara otomatis menyesuaikan sesuai kebutuhan.
- AssociatedRoles – Susunan objek DBClusterPeran.

  Menyediakan daftar Amazon Identity and Access Management (IAM) role yang terkait dengan klaster DB. IAM role yang terkait dengan klaster DB memberikan izin pada klaster DB untuk mengakses layanan Amazon lain atas nama Anda.
- AutomaticRestartTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Waktu di mana klaster DB akan secara otomatis di-restart.
- AvailabilityZones— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance di cluster DB dapat dibuat.
- BacktrackConsumedChangeRecords— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.
- BacktrackWindow— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

Tidak didukung oleh Neptune.

- BackupRetentionPeriod— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.

- Capacity— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- CloneGroupId— String, tipe: `string` (string yang dikodekan UTF-8).

  Mengidentifikasi grup klon yang terkait dengan klaster DB.

- ClusterCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).

- CopyTagsToSnapshot— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke*`true`*, tag akan disalin ke snapshot apa pun dari cluster DB yang dibuat.

- CrossAccountClone— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika diatur ke `true`, klaster DB dapat dikloning di seluruh akun.

- DatabaseName— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama basis data awal klaster DB ini yang disediakan pada waktu membuat, jika salah satu ditentukan ketika klaster DB dibuat. Nama yang sama ini dikembalikan demi kluster DB.

- DBClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk klaster DB.

- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi klaster DB yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi klaster DB.

- DBClusterMembers – Susunan objek DBClusterAnggota.

  Menyediakan daftar instans yang membentuk klaster DB.

- DBClusterParameterGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama grup parameter klaster DB untuk klaster DB ini.

- DbClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi Wilayah Amazon unik yang tetap untuk klaster. Pengidentifikasi ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk cluster DB diakses.
- DBSubnetGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan informasi pada grup subnet yang terkait dengan klaster DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.
- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan.
- EarliestBacktrackTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tidak didukung oleh Neptune.
- EarliestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu paling awal dimana database dapat dipulihkan dengan point-in-time restore.
- EnabledCloudwatchLogsExports— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang cluster DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log. Jenis log yang valid adalah: `audit` (untuk mempublikasikan log audit ke CloudWatch) dan slowquery (untuk mempublikasikan log kueri lambat ke). CloudWatch Lihat Menerbitkan log Neptunus ke log Amazon. CloudWatch
- Endpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan titik akhir koneksi untuk instans primer klaster DB.
- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk klaster DB ini.
- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.

- HostedZoneId— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan ID yang ditetapkan Amazon Route 53 saat Anda membuat zona yang di-hosting.

- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.

- IOOptimizedNextAllowedModificationTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Lain kali Anda dapat memodifikasi cluster DB untuk menggunakan jenis `iopt1` penyimpanan.

- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah BETUL, pengidentifikasi kunci Amazon KMS untuk klaster DB terenkripsi.

- LatestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.

- MultiAZ— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB memiliki instans di beberapa Availability Zone.

- PendingModifiedValues — Sebuah objek ClusterPendingModifiedValues.

  Tipe data ini digunakan sebagai elemen respons dalam `ModifyDBCluster` operasi dan berisi perubahan yang akan diterapkan selama jendela pemeliharaan berikutnya.

- PercentProgress— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan kemajuan operasi sebagai persentase.

- Port— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan oleh mesin basis data.

- PreferredBackupWindow— String, tipe: `string` (string yang dikodekan UTF-8).

Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.

- PreferredMaintenanceWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).

- ReaderEndpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Reader Endpoint untuk klaster DB. Reader Endpoint untuk koneksi menyeimbangkan beban klaster DB di seluruh Replika Baca yang tersedia dalam klaster DB. Saat klien meminta koneksi baru ke Reader Endpoint, Neptune mendistribusikan permintaan koneksi di antara Replika Neptune di klaster DB. Fungsionalitas ini dapat membantu menyeimbangkan beban kerja baca Anda di beberapa Replika Baca di klaster DB Anda.

  Jika terjadi failover, dan Replika Baca yang terhubung dengan Anda dipromosikan menjadi instans primer, koneksi Anda dijatuhkan. Untuk terus mengirimkan beban kerja baca Anda ke Replika Baca lainnya di klaster, Anda kemudian dapat menyambungkan kembali ke Reader Endpoint.

- ReadReplicaIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi Replika Baca terkait dengan klaster DB ini.

- ReplicationSourceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ReplicationType— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ServerlessV2ScalingConfiguration — Sebuah objek ServerlessV2 ScalingConfigurationInfo.

  Menampilkan konfigurasi penskalaan untuk cluster DB Neptunus Tanpa Server.

  Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari klaster DB ini.

- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

Menentukan apakah klaster DB dienkripsi.

- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang digunakan oleh cluster DB.

  Nilai yang Valid:

  - **`standard`**— (default) Menyediakan penyimpanan database hemat biaya untuk aplikasi dengan penggunaan sedang hingga kecil I/O .
  - **`iopt1`**— Memungkinkan penyimpanan I/O-Optimized yang dirancang untuk memenuhi kebutuhan throughput. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    Penyimpanan Neptunus I/O-Optimized hanya tersedia dimulai dengan rilis engine 1.3.0.0.
- VpcSecurityGroups – Susunan objek VpcSecurityGroupMembership.

  Memberikan daftar grup keamanan VPC yang memiliki klaster DB.

Kesalahan

- DBClusterNotFoundFault
- Tidak valid DBCluster StateFault
- Tidak valid DBInstance StateFault

AddRoleToDBCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `add-role-to-db-cluster`

Mengasosiasikan sebuah Identity and Access Management (IAM) role dengan klaster DB Neptune.

Permintaan

- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama kluster DB untuk dikaitkan dengan IAM role.
- FeatureName(dalam CLI:`--feature-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

Nama fitur untuk klaster DB Neptune yang akan dikaitkan dengan IAM role. Untuk daftar nama fitur yang didukung, lihatthe section called “DBEngineVersi”.

- RoleArn(dalam CLI:`--role-arn`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) IAM role yang dikaitkan dengan klaster DB Neptune, misalnya `arn:aws:iam::123456789012:role/NeptuneAccessRole`.

Respons

- Tidak ada parameter Respons.

Kesalahan

- DBClusterNotFoundFault
- DBClusterRoleAlreadyExistsFault
- Tidak valid DBCluster StateFault
- DBClusterRoleQuotaExceededFault

RemoveRoleFromDBCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `remove-role-from-db-cluster`

Membatalkan asosiasi sebuah peran Identity and Access Management (IAM) dari klaster DB.

Permintaan

- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama kluster DB yang akan dihapus kaitannya dengan IAM role.

- FeatureName(dalam CLI:`--feature-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama fitur untuk klaster DB yang akan dihapus kaitannya dengan IAM role. Untuk daftar nama fitur yang didukung, lihatthe section called “Jelaskan DBEngine Versi”.

- RoleArn(dalam CLI:`--role-arn`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) IAM role yang akan dihapus kaitannya dari klaster DB, misalnya `arn:aws:iam::123456789012:role/NeptuneAccessRole`.

Respons

- Tidak ada parameter Respons.

Kesalahan

- DBClusterNotFoundFault
- DBClusterRoleNotFoundFault
- Tidak valid DBCluster StateFault

Failover DBCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `failover-db-cluster`

Memaksakan failover untuk klaster DB.

Failover untuk klaster DB mempromosikan salah satu Replika Baca (instans hanya-baca) di klaster DB menjadi instans primer (penulis klaster).

Amazon Neptune akan secara otomatis melakukan failover ke Replika Baca, jika ada, ketika instans utama gagal. Anda dapat memaksa failover saat ingin mensimulasikan kegagalan instans primer untuk pengujian. Karena tiap instans dalam klaster DB memiliki alamat titik akhirnya sendiri, Anda perlu membersihkan dan membentuk kembali koneksi yang sudah ada yang menggunakan titik akhir tersebut yang ditujukan saat failover selesai.

Permintaan

- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi klaster DB untuk memaksa failover. Parameter ini tidak peka huruf besar kecil.

  Batasan:

  - Harus cocok dengan pengenal yang sudah ada DBCluster.
- TargetDBInstanceIdentifier(dalam CLI:`--target-db-instance-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama instans yang akan dipromosikan ke instans primer.

  Anda harus menentukan pengidentifikasi instans untuk Replika Baca di klaster DB. Misalnya, `mydbcluster-replica1`.

Respons

Berisi detail dari klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam filethe section called “Jelaskan DBClusters”.

- AllocatedStorage— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  `AllocatedStorage` selalu mengembalikan 1, karena ukuran penyimpanan klaster DB Neptune tidak tetap, tetapi justru secara otomatis menyesuaikan sesuai kebutuhan.
- AssociatedRoles – Susunan objek DBClusterPeran.

  Menyediakan daftar Amazon Identity and Access Management (IAM) role yang terkait dengan klaster DB. IAM role yang terkait dengan klaster DB memberikan izin pada klaster DB untuk mengakses layanan Amazon lain atas nama Anda.
- AutomaticRestartTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Waktu di mana klaster DB akan secara otomatis di-restart.
- AvailabilityZones— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance di cluster DB dapat dibuat.
- BacktrackConsumedChangeRecords— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.
- BacktrackWindow— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- BackupRetentionPeriod— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.

- Capacity— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- CloneGroupId— String, tipe: `string` (string yang dikodekan UTF-8).

  Mengidentifikasi grup klon yang terkait dengan klaster DB.

- ClusterCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).

- CopyTagsToSnapshot— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke*`true`*, tag akan disalin ke snapshot apa pun dari cluster DB yang dibuat.

- CrossAccountClone— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika diatur ke `true`, klaster DB dapat dikloning di seluruh akun.

- DatabaseName— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama basis data awal klaster DB ini yang disediakan pada waktu membuat, jika salah satu ditentukan ketika klaster DB dibuat. Nama yang sama ini dikembalikan demi kluster DB.

- DBClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk klaster DB.

- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi klaster DB yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi klaster DB.

- DBClusterMembers – Susunan objek DBClusterAnggota.

  Menyediakan daftar instans yang membentuk klaster DB.

- DBClusterParameterGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama grup parameter klaster DB untuk klaster DB ini.

- DbClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

Pengidentifikasi Wilayah Amazon unik yang tetap untuk klaster. Pengidentifikasi ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk cluster DB diakses.

- DBSubnetGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan informasi pada grup subnet yang terkait dengan klaster DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan.

- EarliestBacktrackTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tidak didukung oleh Neptune.

- EarliestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu paling awal dimana database dapat dipulihkan dengan point-in-time restore.

- EnabledCloudwatchLogsExports— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang cluster DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log. Jenis log yang valid adalah: `audit` (untuk mempublikasikan log audit ke CloudWatch) dan slowquery (untuk mempublikasikan log kueri lambat ke). CloudWatch Lihat Menerbitkan log Neptunus ke log Amazon. CloudWatch

- Endpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan titik akhir koneksi untuk instans primer klaster DB.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk klaster DB ini.

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.

- HostedZoneId— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan ID yang ditetapkan Amazon Route 53 saat Anda membuat zona yang di-hosting.

- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.

- IOOptimizedNextAllowedModificationTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Lain kali Anda dapat memodifikasi cluster DB untuk menggunakan jenis `iopt1` penyimpanan.

- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah BETUL, pengidentifikasi kunci Amazon KMS untuk klaster DB terenkripsi.

- LatestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.

- MultiAZ— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB memiliki instans di beberapa Availability Zone.

- PendingModifiedValues — Sebuah objek ClusterPendingModifiedValues.

  Tipe data ini digunakan sebagai elemen respons dalam `ModifyDBCluster` operasi dan berisi perubahan yang akan diterapkan selama jendela pemeliharaan berikutnya.

- PercentProgress— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan kemajuan operasi sebagai persentase.

- Port— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan oleh mesin basis data.

- PreferredBackupWindow— String, tipe: `string` (string yang dikodekan UTF-8).

Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.

- PreferredMaintenanceWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).

- ReaderEndpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Reader Endpoint untuk klaster DB. Reader Endpoint untuk koneksi menyeimbangkan beban klaster DB di seluruh Replika Baca yang tersedia dalam klaster DB. Saat klien meminta koneksi baru ke Reader Endpoint, Neptune mendistribusikan permintaan koneksi di antara Replika Neptune di klaster DB. Fungsionalitas ini dapat membantu menyeimbangkan beban kerja baca Anda di beberapa Replika Baca di klaster DB Anda.

  Jika terjadi failover, dan Replika Baca yang terhubung dengan Anda dipromosikan menjadi instans primer, koneksi Anda dijatuhkan. Untuk terus mengirimkan beban kerja baca Anda ke Replika Baca lainnya di klaster, Anda kemudian dapat menyambungkan kembali ke Reader Endpoint.

- ReadReplicaIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi Replika Baca terkait dengan klaster DB ini.

- ReplicationSourceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ReplicationType— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ServerlessV2ScalingConfiguration — Sebuah objek ServerlessV2 ScalingConfigurationInfo.

  Menampilkan konfigurasi penskalaan untuk cluster DB Neptunus Tanpa Server.

  Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari klaster DB ini.

- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

Menentukan apakah klaster DB dienkripsi.

- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang digunakan oleh cluster DB.

  Nilai yang Valid:

  - **`standard`**— (default) Menyediakan penyimpanan database hemat biaya untuk aplikasi dengan penggunaan sedang hingga kecil I/O .
  - **`iopt1`**— Memungkinkan penyimpanan I/O-Optimized yang dirancang untuk memenuhi kebutuhan throughput. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    Penyimpanan Neptunus I/O-Optimized hanya tersedia dimulai dengan rilis engine 1.3.0.0.
- VpcSecurityGroups – Susunan objek VpcSecurityGroupMembership.

  Memberikan daftar grup keamanan VPC yang memiliki klaster DB.

Kesalahan

- DBClusterNotFoundFault
- Tidak valid DBCluster StateFault
- Tidak valid DBInstance StateFault

PromoteReadReplicaDBCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `promote-read-replica-db-cluster`

Tidak didukung.

Permintaan

- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung.

Respons

Berisi detail dari klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam filethe section called “Jelaskan DBClusters”.

- AllocatedStorage— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  `AllocatedStorage` selalu mengembalikan 1, karena ukuran penyimpanan klaster DB Neptune tidak tetap, tetapi justru secara otomatis menyesuaikan sesuai kebutuhan.
- AssociatedRoles – Susunan objek DBClusterPeran.

  Menyediakan daftar Amazon Identity and Access Management (IAM) role yang terkait dengan klaster DB. IAM role yang terkait dengan klaster DB memberikan izin pada klaster DB untuk mengakses layanan Amazon lain atas nama Anda.
- AutomaticRestartTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Waktu di mana klaster DB akan secara otomatis di-restart.
- AvailabilityZones— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance di cluster DB dapat dibuat.
- BacktrackConsumedChangeRecords— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.
- BacktrackWindow— a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.
- BackupRetentionPeriod— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.
- Capacity— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Tidak didukung oleh Neptune.
- CloneGroupId— String, tipe: `string` (string yang dikodekan UTF-8).

  Mengidentifikasi grup klon yang terkait dengan klaster DB.
- ClusterCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).

- CopyTagsToSnapshot— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke*`true`*, tag akan disalin ke snapshot apa pun dari cluster DB yang dibuat.

- CrossAccountClone— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika diatur ke `true`, klaster DB dapat dikloning di seluruh akun.

- DatabaseName— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama basis data awal klaster DB ini yang disediakan pada waktu membuat, jika salah satu ditentukan ketika klaster DB dibuat. Nama yang sama ini dikembalikan demi kluster DB.

- DBClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk klaster DB.

- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi klaster DB yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi klaster DB.

- DBClusterMembers – Susunan objek DBClusterAnggota.

  Menyediakan daftar instans yang membentuk klaster DB.

- DBClusterParameterGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama grup parameter klaster DB untuk klaster DB ini.

- DbClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi Wilayah Amazon unik yang tetap untuk klaster. Pengidentifikasi ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk cluster DB diakses.

- DBSubnetGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan informasi pada grup subnet yang terkait dengan klaster DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan.

- EarliestBacktrackTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tidak didukung oleh Neptune.
- EarliestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu paling awal dimana database dapat dipulihkan dengan point-in-time restore.
- EnabledCloudwatchLogsExports— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang cluster DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log. Jenis log yang valid adalah: `audit` (untuk mempublikasikan log audit ke CloudWatch) dan slowquery (untuk mempublikasikan log kueri lambat ke). CloudWatch Lihat Menerbitkan log Neptunus ke log Amazon. CloudWatch
- Endpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan titik akhir koneksi untuk instans primer klaster DB.
- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk klaster DB ini.
- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.
- HostedZoneId— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan ID yang ditetapkan Amazon Route 53 saat Anda membuat zona yang di-hosting.
- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.
- IOOptimizedNextAllowedModificationTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

Lain kali Anda dapat memodifikasi cluster DB untuk menggunakan jenis `iopt1` penyimpanan.

- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah BETUL, pengidentifikasi kunci Amazon KMS untuk klaster DB terenkripsi.

- LatestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.

- MultiAZ— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB memiliki instans di beberapa Availability Zone.

- PendingModifiedValues — Sebuah objek ClusterPendingModifiedValues.

  Tipe data ini digunakan sebagai elemen respons dalam `ModifyDBCluster` operasi dan berisi perubahan yang akan diterapkan selama jendela pemeliharaan berikutnya.

- PercentProgress— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan kemajuan operasi sebagai persentase.

- Port— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan oleh mesin basis data.

- PreferredBackupWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.

- PreferredMaintenanceWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).

- ReaderEndpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Reader Endpoint untuk klaster DB. Reader Endpoint untuk koneksi menyeimbangkan beban klaster DB di seluruh Replika Baca yang tersedia dalam klaster DB. Saat klien meminta koneksi baru ke Reader Endpoint, Neptune mendistribusikan permintaan koneksi di antara Replika Neptune di klaster DB. Fungsionalitas ini dapat membantu menyeimbangkan beban kerja baca Anda di beberapa Replika Baca di klaster DB Anda.

Jika terjadi failover, dan Replika Baca yang terhubung dengan Anda dipromosikan menjadi instans primer, koneksi Anda dijatuhkan. Untuk terus mengirimkan beban kerja baca Anda ke Replika Baca lainnya di klaster, Anda kemudian dapat menyambungkan kembali ke Reader Endpoint.

- ReadReplicaIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi Replika Baca terkait dengan klaster DB ini.

- ReplicationSourceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ReplicationType— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ServerlessV2ScalingConfiguration — Sebuah objek ServerlessV2 ScalingConfigurationInfo.

  Menampilkan konfigurasi penskalaan untuk cluster DB Neptunus Tanpa Server.

  Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari klaster DB ini.

- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB dienkripsi.

- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang digunakan oleh cluster DB.

  Nilai yang Valid:

  - **`standard`**— (default) Menyediakan penyimpanan database hemat biaya untuk aplikasi dengan penggunaan sedang hingga kecil I/O .
  - **`iopt1`**— Memungkinkan penyimpanan I/O-Optimized yang dirancang untuk memenuhi kebutuhan throughput. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    Penyimpanan Neptunus I/O-Optimized hanya tersedia dimulai dengan rilis engine 1.3.0.0.

- VpcSecurityGroups – Susunan objek VpcSecurityGroupMembership.

Memberikan daftar grup keamanan VPC yang memiliki klaster DB.

Kesalahan

- DBClusterNotFoundFault
- Tidak valid DBCluster StateFault

Jelaskan DBClusters (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-db-clusters`

Mengembalikan informasi tentang klaster DB yang disediakan, dan mendukung pemberian nomor halaman.

> **Note**
>
> Operasi ini juga dapat mengembalikan informasi untuk klaster Amazon RDS dan klaster Amazon DocDB.

Permintaan

- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi klaster DB yang disediakan pengguna. Jika parameter ini ditentukan, informasi hanya dari klaster DB tertentu yang dikembalikan. Parameter ini tidak peka huruf besar-kecil.

  Batasan:

  - Jika disediakan, harus cocok dengan DBCluster Identifier yang ada.
- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Filter yang menentukan satu atau lebih klaster DB untuk dideskripsikan.

  Filter yang didukung:

  - `db-cluster-id`- Menerima pengidentifikasi cluster DB dan DB cluster Amazon Resource Names ()ARNs. Daftar hasil hanya akan mencakup informasi tentang cluster DB yang diidentifikasi oleh ini ARNs.

- `engine` - Menerima nama mesin (seperti `neptune`), dan membatasi daftar hasil untuk klaster DB yang dibuat oleh mesin tersebut.

Misalnya, untuk memanggil API ini dari Amazon CLI dan memfilter sehingga hanya klaster DB Neptune yang dikembalikan, Anda bisa menggunakan perintah berikut:

Example

```
aws neptune describe-db-clusters \
            --filters  Name=engine,Values=neptune
```

- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan the section called “Jelaskan DBClusters” sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

  Default: 100

  Kendala: Minimal 20, maksimum 100.

Respons

- DBClusters – Susunan objek DBCluster.

  Berisi daftar klaster DB untuk pengguna.

- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination yang dapat digunakan dalam DBClusters permintaan Deskripsikan berikutnya.

Kesalahan

- DBClusterNotFoundFault

Struktur:

DBCluster (struktur)

Berisi detail dari klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam filethe section called “Jelaskan DBClusters”.

Bidang

- AllocatedStorage— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  `AllocatedStorage` selalu mengembalikan 1, karena ukuran penyimpanan klaster DB Neptune tidak tetap, tetapi justru secara otomatis menyesuaikan sesuai kebutuhan.
- AssociatedRoles— Ini adalah Array DBClusterPeran objek.

  Menyediakan daftar Amazon Identity and Access Management (IAM) role yang terkait dengan klaster DB. IAM role yang terkait dengan klaster DB memberikan izin pada klaster DB untuk mengakses layanan Amazon lain atas nama Anda.
- AutomaticRestartTime— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Waktu di mana klaster DB akan secara otomatis di-restart.
- AvailabilityZones— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance di cluster DB dapat dibuat.
- BacktrackConsumedChangeRecords— Ini adalah LongOptional, dari tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.
- BacktrackWindow— Ini adalah LongOptional, dari tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.
- BackupRetentionPeriod— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.
- Capacity— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

Tidak didukung oleh Neptune.

- CloneGroupId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Mengidentifikasi grup klon yang terkait dengan klaster DB.

- ClusterCreateTime— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).

- CopyTagsToSnapshot— Ini adalah BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke*`true`*, tag akan disalin ke snapshot apa pun dari cluster DB yang dibuat.

- CrossAccountClone— Ini adalah BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika diatur ke `true`, klaster DB dapat dikloning di seluruh akun.

- DatabaseName— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama basis data awal klaster DB ini yang disediakan pada waktu membuat, jika salah satu ditentukan ketika klaster DB dibuat. Nama yang sama ini dikembalikan demi kluster DB.

- DBClusterArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk klaster DB.

- DBClusterIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi klaster DB yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi klaster DB.

- DBClusterMembers— Ini adalah Array DBClusterAnggota objek.

  Menyediakan daftar instans yang membentuk klaster DB.

- DBClusterParameterGroup— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama grup parameter klaster DB untuk klaster DB ini.

- DbClusterResourceId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi Wilayah Amazon unik yang tetap untuk klaster. Pengidentifikasi ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk cluster DB diakses.

- DBSubnetGroup— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan informasi pada grup subnet yang terkait dengan klaster DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.
- DeletionProtection— Ini adalah BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan.
- EarliestBacktrackTime— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tidak didukung oleh Neptune.
- EarliestRestorableTime— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu paling awal dimana database dapat dipulihkan dengan point-in-time restore.
- EnabledCloudwatchLogsExports— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang cluster DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log. Jenis log yang valid adalah: `audit` (untuk mempublikasikan log audit ke CloudWatch) dan slowquery (untuk mempublikasikan log kueri lambat ke). CloudWatch Lihat Menerbitkan log Neptunus ke log Amazon. CloudWatch
- Endpoint— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan titik akhir koneksi untuk instans primer klaster DB.
- Engine— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk klaster DB ini.
- EngineVersion— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.
- GlobalClusterIdentifier— Ini adalah GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.

- HostedZoneId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan ID yang ditetapkan Amazon Route 53 saat Anda membuat zona yang di-hosting.
- IAMDatabaseAuthenticationEnabled— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.
- IOOptimizedNextAllowedModificationTime— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Lain kali Anda dapat memodifikasi cluster DB untuk menggunakan jenis `iopt1` penyimpanan.
- KmsKeyId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah BETUL, pengidentifikasi kunci Amazon KMS untuk klaster DB terenkripsi.
- LatestRestorableTime— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.
- MultiAZ— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB memiliki instans di beberapa Availability Zone.
- PendingModifiedValuesIni adalah sebuah ClusterPendingModifiedValues objek.

  Tipe data ini digunakan sebagai elemen respons dalam `ModifyDBCluster` operasi dan berisi perubahan yang akan diterapkan selama jendela pemeliharaan berikutnya.
- PercentProgress— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan kemajuan operasi sebagai persentase.
- Port— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan oleh mesin basis data.
- PreferredBackupWindow— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.
- PreferredMaintenanceWindow— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).

- ReaderEndpoint— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Reader Endpoint untuk klaster DB. Reader Endpoint untuk koneksi menyeimbangkan beban klaster DB di seluruh Replika Baca yang tersedia dalam klaster DB. Saat klien meminta koneksi baru ke Reader Endpoint, Neptune mendistribusikan permintaan koneksi di antara Replika Neptune di klaster DB. Fungsionalitas ini dapat membantu menyeimbangkan beban kerja baca Anda di beberapa Replika Baca di klaster DB Anda.

  Jika terjadi failover, dan Replika Baca yang terhubung dengan Anda dipromosikan menjadi instans primer, koneksi Anda dijatuhkan. Untuk terus mengirimkan beban kerja baca Anda ke Replika Baca lainnya di klaster, Anda kemudian dapat menyambungkan kembali ke Reader Endpoint.

- ReadReplicaIdentifiers— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi Replika Baca terkait dengan klaster DB ini.

- ReplicationSourceIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ReplicationType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ServerlessV2ScalingConfigurationIni adalah sebuah ServerlessV2 ScalingConfigurationInfo objek.

  Menampilkan konfigurasi penskalaan untuk cluster DB Neptunus Tanpa Server.

  Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- Status— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari klaster DB ini.

- StorageEncrypted— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB dienkripsi.

- StorageType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang digunakan oleh cluster DB.

Nilai yang Valid:

- **`standard`**— (default) Menyediakan penyimpanan database hemat biaya untuk aplikasi dengan penggunaan sedang hingga kecil I/O .
- **`iopt1`**— Memungkinkan penyimpanan I/O-Optimized yang dirancang untuk memenuhi kebutuhan throughput. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

  Penyimpanan Neptunus I/O-Optimized hanya tersedia dimulai dengan rilis engine 1.3.0.0.

- VpcSecurityGroups— Ini adalah Array VpcSecurityGroupMembership objek.

  Memberikan daftar grup keamanan VPC yang memiliki klaster DB.

`DBCluster` digunakan sebagai elemen respon untuk:

- Buat DBCluster
- Hapus DBCluster
- Kegagalan DBCluster
- Memodifikasi DBCluster
- PromoteReadReplicaDBCluster
- Kembalikan DBCluster FromSnapshot
- Kembalikan DBCluster ToPointInTime
- Mulai DBCluster
- Berhenti DBCluster

DBClusterAnggota (struktur)

Berisi informasi tentang sebuah instans yang merupakan bagian dari sebuah klaster DB.

Bidang

- DBClusterParameterGroupStatus— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status grup parameter klaster DB untuk anggota klaster DB ini.

- DBInstanceIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

Menentukan pengidentifikasi instans untuk anggota klaster DB ini.

- IsClusterWriter— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Sebuah nilai yaitu `true` jika anggota klaster adalah instans primer untuk klaster DB dan `false` sebaliknya.

- PromotionTier— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Nilai yang menentukan urutan di mana Replika Baca dipromosikan ke instans primer setelah kegagalan instans primer yang ada.

DBClusterPeran (struktur)

Menyediakan daftar peran Amazon Identity and Access Management (IAM) yang terkait dengan klaster DB.

Bidang

- FeatureName— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama fitur yang terkait dengan peran Amazon Identity and Access Management (IAM). Untuk daftar nama fitur yang didukung, lihatthe section called “Jelaskan DBEngine Versi”.

- RoleArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) dari IAM role yang dikaitkan dengan klaster DB.

- Status— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menjelaskan status keterkaitan antara IAM role dan klaster DB. Properti Status mengembalikan salah satu nilai berikut:

  - `ACTIVE` - ARN IAM role dikaitkan dengan klaster DB dan dapat digunakan untuk mengakses layanan Amazon lain atas nama Anda.
  - `PENDING` - ARN IAM role sedang dikaitkan dengan klaster DB.
  - `INVALID` - ARN IAM role dikaitkan dengan klaster DB, tetapi klaster DB tidak dapat mengasumsikan IAM role untuk mengakses layanan Amazon lain atas nama Anda.

CloudwatchLogsExportConfiguration (struktur)

Pengaturan konfigurasi untuk jenis log yang akan diaktifkan untuk ekspor ke CloudWatch Log untuk instans DB atau cluster DB tertentu.

`DisableLogTypes`Array `EnableLogTypes` dan menentukan log mana yang akan diekspor (atau tidak diekspor) ke Log. CloudWatch

Jenis log yang valid adalah: `audit` (untuk mempublikasikan log audit) dan `slowquery` (untuk menerbitkan log kueri lambat). Lihat Menerbitkan log Neptunus ke log Amazon. CloudWatch

Bidang

- DisableLogTypes— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang akan dinonaktifkan.
- EnableLogTypes— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang akan diaktifkan.

PendingCloudwatchLogsExports (struktur)

Daftar jenis log yang konfigurasinya masih tertunda. Dengan kata lain, jenis log ini sedang dalam proses diaktifkan atau dinonaktifkan.

Jenis log yang valid adalah: `audit` (untuk mempublikasikan log audit) dan `slowquery` (untuk menerbitkan log kueri lambat). Lihat Menerbitkan log Neptunus ke log Amazon. CloudWatch

Bidang

- LogTypesToDisable— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis log yang sedang dalam proses diaktifkan. Setelah diaktifkan, jenis log ini diekspor ke CloudWatch Log.
- LogTypesToEnable— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis log yang sedang dalam proses dinonaktifkan. Setelah dinonaktifkan, jenis log ini tidak diekspor ke CloudWatch Log.

ClusterPendingModifiedValues (struktur)

Tipe data ini digunakan sebagai elemen respons dalam `ModifyDBCluster` operasi dan berisi perubahan yang akan diterapkan selama jendela pemeliharaan berikutnya.

Bidang

- AllocatedStorage— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Ukuran penyimpanan yang dialokasikan dalam gibibytes (GiB) untuk mesin database. Untuk Neptunus`AllocatedStorage`, selalu mengembalikan 1, karena ukuran penyimpanan cluster DB Neptunus tidak tetap, melainkan secara otomatis menyesuaikan sesuai kebutuhan.
- BackupRetentionPeriod— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah hari di mana snapshot DB otomatis dipertahankan.
- DBClusterIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nilai DBCluster Identifier untuk cluster DB.
- EngineVersion— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Versi mesin database.
- IAMDatabaseAuthenticationEnabled— Ini adalah BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang menunjukkan apakah pemetaan akun AWS Identity and Access Management (IAM) ke akun database diaktifkan.
- Iops— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Nilai IOPS yang Disediakan (operasi I/O per detik). Pengaturan ini hanya untuk cluster DB multi-AZ.
- PendingCloudwatchLogsExportsIni adalah sebuah PendingCloudwatchLogsExports objek.

  `PendingCloudwatchLogsExports`Struktur ini menentukan perubahan yang tertunda untuk CloudWatch log mana yang diaktifkan dan mana yang dinonaktifkan.
- StorageType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Perubahan tertunda dalam jenis penyimpanan untuk cluster DB. Nilai yang Valid:

- **`standard`**— (default) Mengkonfigurasi penyimpanan database hemat biaya untuk aplikasi dengan penggunaan sedang hingga kecil. I/O
- **`iopt1`**— Memungkinkan penyimpanan I/O-Optimized yang dirancang untuk memenuhi kebutuhan throughput. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

  Penyimpanan Neptunus I/O-Optimized hanya tersedia dimulai dengan rilis engine 1.3.0.0.

API Database Global Neptunus

Tindakan:

- CreateGlobalCluster (tindakan)
- DeleteGlobalCluster (tindakan)
- ModifyGlobalCluster (tindakan)
- DescribeGlobalClusters (tindakan)
- FailoverGlobalCluster (tindakan)
- RemoveFromGlobalCluster (tindakan)

Struktur:

- GlobalCluster (struktur)
- GlobalClusterMember (struktur)

CreateGlobalCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `create-global-cluster`

Membuat database global Neptunus yang tersebar di beberapa Wilayah Amazon. Basis data global berisi cluster primer tunggal dengan kemampuan baca-tulis, dan cluster sekunder read-only yang menerima data dari cluster primer melalui replikasi kecepatan tinggi yang dilakukan oleh subsistem penyimpanan Neptunus.

Anda dapat membuat database global yang awalnya kosong, dan kemudian menambahkan cluster primer dan cluster sekunder ke dalamnya, atau Anda dapat menentukan cluster Neptunus yang ada selama operasi create untuk menjadi cluster utama database global.

Permintaan

- DatabaseName(dalam CLI:`--database-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama untuk database global baru (hingga 64 karakter alfa-numerik.
- DeletionProtection(dalam CLI:`--deletion-protection`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Pengaturan perlindungan penghapusan untuk basis data global baru. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan.
- Engine(dalam CLI:`--engine`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama mesin database yang akan digunakan dalam database global.

  Nilai yang valid: `neptune`
- EngineVersion(dalam CLI:`--engine-version`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Versi mesin Neptunus akan digunakan oleh database global.

  Nilai yang valid: `1.2.0.0` atau di atas.
- GlobalClusterIdentifier(dalam CLI:`--global-cluster-identifier`) - Wajib: a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Cluster identifier dari cluster database global baru.
- SourceDBClusterIdentifier(dalam CLI:`--source-db-cluster-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  (Opsional) Nama Sumber Daya Amazon (ARN) dari kluster DB Neptunus yang ada untuk digunakan sebagai cluster utama database global baru.
- StorageEncrypted(dalam CLI:`--storage-encrypted`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Pengaturan enkripsi penyimpanan untuk cluster database global baru.

Respons

Berisi rincian database global Amazon Neptunus.

Tipe data ini digunakan sebagai elemen respons untukthe section called “CreateGlobalCluster”,,the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”the section called “DeleteGlobalCluster”,the section called “FailoverGlobalCluster”, dan the section called “RemoveFromGlobalCluster” tindakan.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Pengaturan perlindungan penghapusan untuk database global.
- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Mesin database Neptunus yang digunakan oleh database global (). `"neptune"`
- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Versi mesin Neptunus yang digunakan oleh database global.
- GlobalClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) untuk database global.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.
- GlobalClusterMembers – Susunan objek GlobalClusterMember.

  Daftar cluster ARNs dan instance ARNs untuk semua cluster DB yang merupakan bagian dari database global.
- GlobalClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi yang tidak dapat diubah untuk database global yang unik di semua wilayah. Pengidentifikasi ini ditemukan di entri CloudTrail log setiap kali kunci KMS untuk cluster DB diakses.
- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan keadaan saat ini dari database global ini.
- StorageEncrypted— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

Pengaturan enkripsi penyimpanan untuk database global.

Kesalahan

- GlobalClusterAlreadyExistsFault
- GlobalClusterQuotaExceededFault
- Tidak valid DBCluster StateFault
- DBClusterNotFoundFault

DeleteGlobalCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `delete-global-cluster`

Menghapus database global. Cluster primer dan semua sekunder harus sudah terlepas atau dihapus terlebih dahulu.

Permintaan

- GlobalClusterIdentifier(dalam CLI:`--global-cluster-identifier`) - Wajib: a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Cluster identifier dari cluster database global sedang dihapus.

Respons

Berisi rincian database global Amazon Neptunus.

Tipe data ini digunakan sebagai elemen respons untukthe section called “CreateGlobalCluster”,,the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”the section called “DeleteGlobalCluster”,the section called “FailoverGlobalCluster”, dan the section called “RemoveFromGlobalCluster” tindakan.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Pengaturan perlindungan penghapusan untuk database global.
- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

Mesin database Neptunus yang digunakan oleh database global (). `"neptune"`

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Versi mesin Neptunus yang digunakan oleh database global.

- GlobalClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) untuk database global.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.

- GlobalClusterMembers – Susunan objek GlobalClusterMember.

  Daftar cluster ARNs dan instance ARNs untuk semua cluster DB yang merupakan bagian dari database global.

- GlobalClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi yang tidak dapat diubah untuk database global yang unik di semua wilayah. Pengidentifikasi ini ditemukan di entri CloudTrail log setiap kali kunci KMS untuk cluster DB diakses.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan keadaan saat ini dari database global ini.

- StorageEncrypted— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Pengaturan enkripsi penyimpanan untuk database global.

Kesalahan

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

ModifyGlobalCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `modify-global-cluster`

Ubah pengaturan untuk klaster global Amazon Neptunus. Anda dapat mengubah satu atau beberapa parameter konfigurasi database dengan menentukan parameter ini dan nilai barunya dalam permintaan.

Permintaan

- AllowMajorVersionUpgrade(dalam CLI:`--allow-major-version-upgrade`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang menunjukkan apakah pembaruan versi utama diizinkan.

  Kendala: Anda harus mengizinkan upgrade versi mayor jika Anda menentukan nilai untuk `EngineVersion` parameter yang merupakan versi mayor yang berbeda dari versi cluster DB saat ini.

  Jika Anda memutakhirkan versi utama database global, grup parameter instans cluster dan DB disetel ke grup parameter default untuk versi baru, jadi Anda harus menerapkan grup parameter kustom apa pun setelah menyelesaikan pemutakhiran.
- DeletionProtection(dalam CLI:`--deletion-protection`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah database global telah mengaktifkan perlindungan penghapusan. Database global tidak dapat dihapus ketika perlindungan penghapusan diaktifkan.
- EngineVersion(dalam CLI:`--engine-version`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nomor versi mesin basis data yang ingin Anda tingkatkan. Mengubah parameter ini akan mengakibatkan pemadaman. Perubahan diterapkan selama jendela pemeliharaan berikutnya kecuali `ApplyImmediately` diaktifkan.

  Untuk mencantumkan semua versi mesin Neptunus yang tersedia, gunakan perintah berikut:

  Example

```
aws neptune describe-db-engine-versions \
          --engine neptune \
```

```
    --query '*[]|[?SupportsGlobalDatabases == 'true'].[EngineVersion]'
```

- GlobalClusterIdentifier(dalam CLI:`--global-cluster-identifier`) - Wajib: a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Pengidentifikasi cluster DB untuk cluster global sedang dimodifikasi. Parameter ini tidak peka huruf besar kecil.

  Constraints: Harus cocok dengan identifier dari cluster database global yang ada.

- NewGlobalClusterIdentifier(dalam CLI:`--new-global-cluster-identifier`) — a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Sebuah identifier cluster baru untuk menetapkan ke database global. Nilai ini disimpan sebagai string huruf kecil.

  Kendala:

  - Harus berisi 1 hingga 63 huruf, angka, atau tanda hubung.
  - Karakter pertama harus berupa huruf.
  - Tidak dapat diakhiri dengan sebuah tanda hubung atau mengandung dua tanda hubung berturut-turut

  Contoh: `my-cluster2`

Respons

Berisi rincian database global Amazon Neptunus.

Tipe data ini digunakan sebagai elemen respons untukthe section called “CreateGlobalCluster”,,the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”the section called “DeleteGlobalCluster”,the section called “FailoverGlobalCluster”, dan the section called “RemoveFromGlobalCluster” tindakan.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Pengaturan perlindungan penghapusan untuk database global.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

Mesin database Neptunus yang digunakan oleh database global (). `"neptune"`

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Versi mesin Neptunus yang digunakan oleh database global.

- GlobalClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) untuk database global.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.

- GlobalClusterMembers – Susunan objek GlobalClusterMember.

  Daftar cluster ARNs dan instance ARNs untuk semua cluster DB yang merupakan bagian dari database global.

- GlobalClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi yang tidak dapat diubah untuk database global yang unik di semua wilayah. Pengidentifikasi ini ditemukan di entri CloudTrail log setiap kali kunci KMS untuk cluster DB diakses.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan keadaan saat ini dari database global ini.

- StorageEncrypted— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Pengaturan enkripsi penyimpanan untuk database global.

Kesalahan

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

DescribeGlobalClusters (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-global-clusters`

Mengembalikan informasi tentang cluster database global Neptunus. API ini mendukung pagination (pemberian nomor halaman).

Permintaan

- GlobalClusterIdentifier(dalam CLI:`--global-cluster-identifier`) — a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Pengidentifikasi klaster DB yang disediakan pengguna. Jika parameter ini ditentukan, hanya informasi tentang cluster DB yang ditentukan dikembalikan. Parameter ini tidak peka huruf besar kecil.

  Kendala: Jika disediakan, harus cocok dengan pengidentifikasi cluster DB yang ada.
- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  (Opsional) Token pagination yang dikembalikan oleh panggilan sebelumnya ke`DescribeGlobalClusters`. Jika parameter ini ditentukan, respons hanya akan menyertakan catatan di luar penanda, hingga nomor yang ditentukan oleh`MaxRecords`.
- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika ada lebih banyak catatan daripada `MaxRecords` nilai yang ditentukan, token penanda pagination disertakan dalam respons yang dapat Anda gunakan untuk mengambil hasil yang tersisa.

  Default: `100`

  Kendala: Minimal 20, maksimum 100.

Respons

- GlobalClusters – Susunan objek GlobalCluster.

  Daftar cluster global dan instance yang dikembalikan oleh permintaan ini.
- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

Token pagination. Jika parameter ini dikembalikan dalam respons, lebih banyak catatan tersedia, yang dapat diambil oleh satu atau lebih panggilan tambahan ke`DescribeGlobalClusters`.

Kesalahan

- GlobalClusterNotFoundFault

FailoverGlobalCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `failover-global-cluster`

Memulai proses failover untuk database global Neptunus.

Failover untuk database global Neptunus mempromosikan salah satu cluster DB hanya-baca sekunder menjadi cluster DB primer dan mendemotasi cluster DB primer menjadi cluster DB sekunder (hanya-baca). Dengan kata lain, peran cluster DB primer saat ini dan cluster DB sekunder target yang dipilih diaktifkan. Cluster DB sekunder yang dipilih mengasumsikan read/write kemampuan penuh untuk database global Neptunus.

> **Note**
>
> Tindakan ini hanya berlaku untuk database global Neptunus. Tindakan ini hanya dimaksudkan untuk digunakan pada database global Neptunus yang sehat dengan kluster DB Neptunus yang sehat dan tidak ada pemadaman di seluruh wilayah, untuk menguji skenario pemulihan bencana atau untuk mengkonfigurasi ulang topologi database global.

Permintaan

- GlobalClusterIdentifier(dalam CLI:`--global-cluster-identifier`) - Wajib: a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Pengidentifikasi database global Neptunus yang seharusnya gagal. Identifier adalah kunci unik yang ditetapkan oleh pengguna ketika database global Neptunus dibuat. Dengan kata lain, itu adalah nama database global yang ingin Anda gagal.

  Kendala: Harus cocok dengan pengenal database global Neptunus yang ada.

- TargetDbClusterIdentifier(dalam CLI:`--target-db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) dari cluster DB Neptunus sekunder yang ingin Anda promosikan ke primer untuk database global.

Respons

Berisi rincian database global Amazon Neptunus.

Tipe data ini digunakan sebagai elemen respons untukthe section called “CreateGlobalCluster”,,the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”the section called “DeleteGlobalCluster”,the section called “FailoverGlobalCluster”, dan the section called “RemoveFromGlobalCluster” tindakan.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Pengaturan perlindungan penghapusan untuk database global.
- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Mesin database Neptunus yang digunakan oleh database global (). `"neptune"`
- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Versi mesin Neptunus yang digunakan oleh database global.
- GlobalClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) untuk database global.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.
- GlobalClusterMembers – Susunan objek GlobalClusterMember.

  Daftar cluster ARNs dan instance ARNs untuk semua cluster DB yang merupakan bagian dari database global.
- GlobalClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

Pengidentifikasi yang tidak dapat diubah untuk database global yang unik di semua wilayah. Pengidentifikasi ini ditemukan di entri CloudTrail log setiap kali kunci KMS untuk cluster DB diakses.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan keadaan saat ini dari database global ini.

- StorageEncrypted— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Pengaturan enkripsi penyimpanan untuk database global.

Kesalahan

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault
- Tidak valid DBCluster StateFault
- DBClusterNotFoundFault

RemoveFromGlobalCluster (tindakan)

Nama AWS CLI untuk API ini adalah:. `remove-from-global-cluster`

Melepaskan cluster DB Neptunus dari database global Neptunus. Cluster sekunder menjadi cluster mandiri normal dengan kemampuan baca-tulis alih-alih hanya-baca, dan tidak lagi menerima data dari cluster utama.

Permintaan

- DbClusterIdentifier(dalam CLI:`--db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) mengidentifikasi cluster yang akan terlepas dari cluster database global Neptunus.

- GlobalClusterIdentifier(dalam CLI:`--global-cluster-identifier`) - Wajib: a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

Pengidentifikasi database global Neptunus untuk melepaskan cluster DB Neptunus yang ditentukan.

Respons

Berisi rincian database global Amazon Neptunus.

Tipe data ini digunakan sebagai elemen respons untukthe section called “CreateGlobalCluster”,,the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”the section called “DeleteGlobalCluster”,the section called “FailoverGlobalCluster”, dan the section called “RemoveFromGlobalCluster” tindakan.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Pengaturan perlindungan penghapusan untuk database global.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Mesin database Neptunus yang digunakan oleh database global (). `"neptune"`

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Versi mesin Neptunus yang digunakan oleh database global.

- GlobalClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) untuk database global.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.

- GlobalClusterMembers – Susunan objek GlobalClusterMember.

  Daftar cluster ARNs dan instance ARNs untuk semua cluster DB yang merupakan bagian dari database global.

- GlobalClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

Pengidentifikasi yang tidak dapat diubah untuk database global yang unik di semua wilayah. Pengidentifikasi ini ditemukan di entri CloudTrail log setiap kali kunci KMS untuk cluster DB diakses.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan keadaan saat ini dari database global ini.

- StorageEncrypted— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Pengaturan enkripsi penyimpanan untuk database global.

Kesalahan

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault
- DBClusterNotFoundFault

Struktur:

GlobalCluster (struktur)

Berisi rincian database global Amazon Neptunus.

Tipe data ini digunakan sebagai elemen respons untukthe section called “CreateGlobalCluster”,,the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”the section called “DeleteGlobalCluster”,the section called “FailoverGlobalCluster”, dan the section called “RemoveFromGlobalCluster” tindakan.

Bidang

- DeletionProtection— Ini adalah BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Pengaturan perlindungan penghapusan untuk database global.

- Engine— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Mesin database Neptunus yang digunakan oleh database global (). `"neptune"`

- EngineVersion— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

Versi mesin Neptunus yang digunakan oleh database global.

- GlobalClusterArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) untuk database global.

- GlobalClusterIdentifier— Ini adalah GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.

- GlobalClusterMembers— Ini adalah Array GlobalClusterMember objek.

  Daftar cluster ARNs dan instance ARNs untuk semua cluster DB yang merupakan bagian dari database global.

- GlobalClusterResourceId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi yang tidak dapat diubah untuk database global yang unik di semua wilayah. Pengidentifikasi ini ditemukan di entri CloudTrail log setiap kali kunci KMS untuk cluster DB diakses.

- Status— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan keadaan saat ini dari database global ini.

- StorageEncrypted— Ini adalah BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Pengaturan enkripsi penyimpanan untuk database global.

`GlobalCluster` digunakan sebagai elemen respons untuk:

- CreateGlobalCluster
- ModifyGlobalCluster
- DeleteGlobalCluster
- RemoveFromGlobalCluster
- FailoverGlobalCluster

GlobalClusterMember (struktur)

Struktur data dengan informasi tentang cluster primer dan sekunder yang terkait dengan database global Neptunus.

Bidang

- DBClusterArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) untuk setiap cluster Neptunus.
- IsWriter— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah cluster Neptunus adalah cluster utama (yaitu, memiliki kemampuan baca-tulis) untuk database global Neptunus yang terkait dengannya.
- Readers— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) untuk setiap cluster sekunder hanya-baca yang terkait dengan database global Neptunus.

API Instans Neptune

Tindakan:

- Buat DBInstance (tindakan)
- Hapus DBInstance (tindakan)
- Memodifikasi DBInstance (tindakan)
- Reboot DBInstance (tindakan)
- Jelaskan DBInstances (tindakan)
- DescribeOrderableDBInstanceOpsi (tindakan)
- DescribeValidDBInstanceModifikasi (tindakan)

Struktur:

- DBInstance (struktur)
- DBInstanceStatusInfo (struktur)
- DBInstanceOpsi yang Dapat Dipesan (struktur)

- PendingModifiedValues (struktur)
- ValidStorageOptions (struktur)
- Valid DBInstance ModificationsMessage (struktur)

Buat DBInstance (tindakan)

Nama AWS CLI untuk API ini adalah:. `create-db-instance`

Menciptakan instans DB baru.

Permintaan

- AutoMinorVersionUpgrade(dalam CLI:`--auto-minor-version-upgrade`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan bahwa peningkatan mesin kecil diterapkan secara otomatis ke instans DB selama jendela pemeliharaan.

  Default: `true`
- AvailabilityZone(dalam CLI:`--availability-zone`) — String, tipe: `string` (string yang dikodekan UTF-8).

  EC2 Availability Zone tempat instans DB dibuat

  Default: Availability Zone yang dipilih sistem secara acak di Wilayah Amazon titik akhir.

  Contoh: `us-east-1d`

  Kendala: AvailabilityZone Parameter tidak dapat ditentukan jika parameter MultiAZ disetel ke. `true` Availability Zone yang ditentukan harus berada dalam Wilayah Amazon yang sama dengan titik akhir saat ini.
- BackupRetentionPeriod(dalam CLI:`--backup-retention-period`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah hari penyimpanan cadangan otomatis.

  Tidak berlaku. Periode retensi untuk backup otomatis dikelola oleh klaster DB. Untuk informasi selengkapnya, lihat the section called “Buat DBCluster”.

  Default: 1

Batasan:

- Harus berupa nilai dari 0 hingga 35
- Tidak dapat diatur ke 0 jika instans DB adalah sumber untuk Replika Baca

- CopyTagsToSnapshot(dalam CLI:`--copy-tags-to-snapshot`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  True untuk menyalin semua tag dari instans DB untuk snapshot instans DB, dan sebaliknya false. Default-nya adalah salah.
- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi klaster DB tempat instans akan berada.

  Untuk informasi tentang pembuatan klaster DB, lihat the section called “Buat DBCluster”.

  Tipe: String
- DBInstanceClass(dalam CLI:`--db-instance-class`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kapasitas komputasi dan memori instans DB; misalnya, `db.m4.large`. Tidak semua kelas instans DB tersedia di semua Wilayah Amazon.
- DBInstanceIdentifier(dalam CLI:`--db-instance-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi instans DB. Parameter ini disimpan sebagai string huruf kecil.

  Batas:

  - Harus berisi 1 sampai 63 huruf, angka, atau tanda hubung.
  - Karakter pertama harus berupa huruf.
  - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.

  Contoh: `mydbinstance`
- DBName(dalam CLI:`--db-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung.
- DBParameterGroupName(dalam CLI:`--db-parameter-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

Nama grup parameter DB yang akan dikaitkan dengan instans DB ini. Jika argumen ini dihilangkan, DBParameter Grup default untuk mesin yang ditentukan digunakan.

Batasan:

- Harus berisi 1 sampai 255 huruf, angka, atau tanda hubung.
- Karakter pertamanya harus berupa huruf
- Tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung yang berurutan

- DBSecurityGroups(dalam CLI:`--db-security-groups`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar grup keamanan DB untuk dikaitkan dengan instans DB ini.

  Default: Grup keamanan DB default untuk mesin basis data.
- DBSubnetGroupName(dalam CLI:`--db-subnet-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Grup subnet DB untuk dihubungkan dengan instans DB ini.

  Jika tidak ada grup subnet DB, maka itu adalah instans DB non-VPC.
- DeletionProtection(dalam CLI:`--deletion-protection`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang menunjukkan apakah instans DB mengaktifkan perlindungan penghapusan. Basis data tidak dapat dihapus saat perlindungan penghapusan diaktifkan. Secara default, perlindungan penghapusan dinonaktifkan. Lihat Menghapus instans DB.

  Instans DB dalam klaster DB dapat dihapus bahkan ketika perlindungan penghapusan diaktifkan dalam klaster DB induk mereka.
- Domain(dalam CLI:`--domain`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Tentukan Domain Direktori Aktif tempat membuat instans.
- DomainIAMRoleName(dalam CLI:`--domain-iam-role-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Tentukan nama IAM role yang akan digunakan saat melakukan panggilan API ke Directory Service.
- EnableCloudwatchLogsExports(dalam CLI:`--enable-cloudwatch-logs-exports`) — String, tipe: `string` (string yang dikodekan UTF-8).

Daftar jenis log yang perlu diaktifkan untuk mengekspor ke CloudWatch Log.

- EnableIAMDatabaseAuthentication(dalam CLI:`--enable-iam-database-authentication`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Tidak didukung oleh Neptune (diabaikan).
- Engine(dalam CLI:`--engine`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama mesin basis data yang akan digunakan untuk instans ini.

  Nilai Valid: `neptune`
- EngineVersion(dalam CLI:`--engine-version`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nomor versi mesin basis data yang akan digunakan. Saat ini, pengaturan parameter ini tidak berpengaruh.
- Iops(dalam CLI:`--iops`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah Provisioned IOPS (operasi input/output per detik) yang akan awalnya dialokasikan untuk instans DB.
- KmsKeyId(dalam CLI:`--kms-key-id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi kunci Amazon KMS untuk instans DB terenkripsi.

  Pengidentifikasi kunci KMS adalah Amazon Resource Name (ARN) untuk kunci enkripsi KMS. Jika Anda membuat instans DB dengan akun Amazon yang sama yang memiliki kunci enkripsi KMS yang digunakan untuk mengenkripsi instans DB baru, maka Anda dapat menggunakan alias kunci KMS alih-alih ARN untuk kunci enkripsi KMS.

  Tidak berlaku. Pengidentifikasi kunci KMS dikelola oleh klaster DB. Untuk informasi selengkapnya, lihat the section called “Buat DBCluster”.

  Jika parameter `StorageEncrypted` true dan Anda tidak menentukan nilai untuk `KmsKeyId`, maka Amazon Neptune akan menggunakan kunci enkripsi default Anda. Amazon KMS menciptakan kunci enkripsi default untuk akun Amazon Anda. Akun Amazon Anda memiliki kunci enkripsi default yang berbeda untuk setiap Wilayah Amazon.
- LicenseModel(dalam CLI:`--license-model`) — String, tipe: `string` (string yang dikodekan UTF-8).

Informasi model lisensi untuk instans DB ini.

Nilai yang valid: `license-included` | `bring-your-own-license` | `general-public-license`

- MonitoringInterval(dalam CLI:`--monitoring-interval`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Interval, dalam detik, antara titik-titik saat metrik Pemantauan yang Ditingkatkan dikumpulkan untuk instans DB. Untuk menonaktifkan pengumpulan metrik Pemantauan yang Ditingkatkan, tentukan 0. Default-nya adalah 0.

  Jika `MonitoringRoleArn` ditentukan, maka Anda juga harus mengatur `MonitoringInterval` ke nilai selain 0.

  Nilai Valid: `0, 1, 5, 10, 15, 30, 60`

- MonitoringRoleArn(dalam CLI:`--monitoring-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN untuk peran IAM yang memungkinkan Neptunus mengirim metrik pemantauan yang ditingkatkan ke Amazon Logs. CloudWatch Misalnya, `arn:aws:iam:123456789012:role/emaccess`.

  Jika `MonitoringInterval` diatur ke nilai selain 0, maka Anda harus menyediakan nilai `MonitoringRoleArn`.

- MultiAZ(dalam CLI:`--multi-az`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan jika instans DB adalah deployment Multi-AZ. Anda tidak dapat mengatur parameter jika AvailabilityZone parameter MultiAZ disetel ke true.

- Port(dalam CLI:`--port`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Nomor port tempat basis data menerima koneksi.

  Tidak berlaku. Port dikelola oleh kluster DB. Untuk informasi selengkapnya, lihat the section called “Buat DBCluster”.

  Default: 8182

Jenis: Integer

- PreferredBackupWindow(dalam CLI:`--preferred-backup-window`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Rentang waktu harian selama backup otomatis dibuat.

  Tidak berlaku. Kisaran waktu harian untuk membuat backup otomatis dikelola oleh klaster DB. Untuk informasi selengkapnya, lihat the section called “Buat DBCluster”.
- PreferredMaintenanceWindow(dalam CLI:`--preferred-maintenance-window`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Rentang waktu setiap minggu untuk melakukan pemeliharaan sistem, dalam Waktu Universal Terkoordinasi (UTC).

  Format: `ddd:hh24:mi-ddd:hh24:mi`

  Default adalah jendela 30 menit yang dipilih secara acak dari blok waktu 8 jam per Wilayah Amazon, yang dilakukan pada sembarang hari dalam seminggu.

  Hari yang valid: Sen, Sel, Rab, Kam, Jum, Sab, Min.

  Kendala: Minimum 30 menit jendela.
- PromotionTier(dalam CLI:`--promotion-tier`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Nilai yang menentukan urutan di mana Replika Baca dipromosikan ke instans primer setelah kegagalan instans primer yang ada.

  Default: 1

  Nilai yang Valid: 0-15
- PubliclyAccessible(dalam CLI:`--publicly-accessible`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Bendera ini seharusnya tidak lagi digunakan.
- StorageEncrypted(dalam CLI:`--storage-encrypted`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah instans DB dienkripsi atau tidak.

Tidak berlaku. Enkripsi untuk instans DB dikelola oleh klaster DB. Untuk informasi selengkapnya, lihat the section called “Buat DBCluster”.

Default: false

- StorageType(dalam CLI:`--storage-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak berlaku. Di Neptunus jenis penyimpanan dikelola pada tingkat DB Cluster.

- Tags(dalam CLI:`--tags`) — Sebuah array objek. Tag

  Tanda yang akan ditetapkan ke instans baru.

- TdeCredentialArn(dalam CLI:`--tde-credential-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari penyimpanan kunci dengan yang terkait dengan instans untuk enkripsi TDE.

- TdeCredentialPassword(dalam CLI:`--tde-credential-password`) — a SensitiveString, tipe: `string` (string yang dikodekan UTF-8).

  Kata sandi untuk ARN yang diberikan dari penyimpanan kunci untuk mengakses perangkat.

- Timezone(dalam CLI:`--timezone`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Zona waktu instans DB.

- VpcSecurityGroupIds(dalam CLI:`--vpc-security-group-ids`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar grup keamanan EC2 VPC untuk dikaitkan dengan instans DB ini.

  Tidak berlaku. Daftar terkait grup keamanan EC2 VPC dikelola oleh cluster DB. Untuk informasi selengkapnya, lihat the section called “Buat DBCluster”.

  Default: Grup keamanan EC2 VPC default untuk VPC grup subnet DB.

Respons

Berisi detail dari instans DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBInstances”.

- AutoMinorVersionUpgrade— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan bahwa patch versi minor diterapkan secara otomatis.

- AvailabilityZone— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama Availability Zone tempat instans DB tersebut berada.

- BackupRetentionPeriod— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.

- CACertificateIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi sertifikat CA untuk instans DB ini.

- CopyTagsToSnapshot— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah tanda disalin dari instans DB ke snapshot instans DB.

- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika instans DB adalah anggota klaster DB, berisi nama klaster DB tempat instans DB tersebut menjadi anggota.

- DBInstanceArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk instans DB.

- DBInstanceClass— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama kelas kapasitas komputasi dan memori instans DB.

- DBInstanceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi basis data yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi sebuah instans DB.

- DbInstancePort— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port tempat instans DB mendengarkan. Jika instans DB adalah bagian dari klaster DB, ini bisa menjadi port yang berbeda dari port klaster DB.

- DBInstanceStatus— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari basis data ini.

- DbiResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

Pengidentifikasi Wilayah Amazon unik yang tetap untuk instans DB. Pengidentifikasi ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk instans DB diakses.

- DBName— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama database.

- DBParameterGroups – Susunan objek DBParameterGroupStatus.

  Menyediakan daftar grup parameter DB yang diterapkan ke instans DB ini.

- DBSecurityGroups – Susunan objek DBSecurityGroupMembership.

  Menyediakan Daftar elemen grup keamanan DB yang hanya berisi subelemen `DBSecurityGroup.Name` dan `DBSecurityGroup.Status`.

- DBSubnetGroup — Sebuah objek DBSubnetKelompok.

  Menentukan informasi tentang grup subnet yang terkait dengan instans DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Instans tidak dapat dihapus saat perlindungan penghapusan diaktifkan. Lihat Menghapus instans DB.

- DomainMemberships – Susunan objek DomainMembership.

  Tidak didukung

- EnabledCloudwatchLogsExports— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang instans DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log.

- Endpoint — Sebuah objek Titik Akhir.

  Menentukan titik akhir koneksi.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk instans DB ini.

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.

- EnhancedMonitoringResourceArn— String, tipe: `string` (string yang dikodekan UTF-8).

Nama Sumber Daya Amazon (ARN) dari aliran CloudWatch log Amazon Logs yang menerima data metrik Pemantauan yang Ditingkatkan untuk instans DB.

- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  True jika autentikasi Amazon Identity and Access Management (IAM) diaktifkan, dan jika sebaliknya akan false.

- InstanceCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menyediakan tanggal dan waktu saat instans DB dibuat.

- Iops— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan nilai Provisioned IOPS (operasi I/O per detik).

- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung: Enkripsi untuk instans DB dikelola oleh klaster DB.

- LatestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.

- LicenseModel— String, tipe: `string` (string yang dikodekan UTF-8).

  Informasi model lisensi untuk instans DB ini.

- MonitoringInterval— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Interval, dalam detik, antara titik-titik saat metrik Pemantauan yang Ditingkatkan dikumpulkan untuk instans DB.

- MonitoringRoleArn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN untuk peran IAM yang memungkinkan Neptunus mengirim metrik Pemantauan yang Ditingkatkan ke Log Amazon. CloudWatch

- MultiAZ— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan jika instans DB adalah deployment Multi-AZ.

- PendingModifiedValues — Sebuah objek PendingModifiedValues.

Menentukan bahwa perubahan pada instans DB sedang tertunda. Elemen ini disertakan hanya ketika perubahan tertunda. Perubahan spesifik diidentifikasi oleh sub elemen.

- PreferredBackupWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.
- PreferredMaintenanceWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).
- PromotionTier— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Nilai yang menentukan urutan di mana Replika Baca dipromosikan ke instans primer setelah kegagalan instans primer yang ada.
- PubliclyAccessible— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Bendera ini seharusnya tidak lagi digunakan.
- ReadReplicaDBClusterIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi klaster DB yang merupakan Replika Baca instans DB ini.
- ReadReplicaDBInstanceIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu pengidentifikasi atau lebih dari Replika Baca yang terkait dengan instans DB ini.
- ReadReplicaSourceDBInstanceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi instans DB sumber jika instans DB ini adalah Replika Baca.
- SecondaryAvailabilityZone— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika ada, menentukan nama Availability Zone sekunder untuk instans DB dengan dukungan multi-AZ.
- StatusInfos – Susunan objek DBInstanceStatusInfo.

  Status Replika Baca. Jika instans bukan Replika Baca, ini kosong.
- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Tidak didukung: Enkripsi untuk instans DB dikelola oleh klaster DB.
- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

Menentukan jenis penyimpanan yang terkait dengan instans DB.

- TdeCredentialArn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari penyimpanan kunci dengan yang terkait dengan instans untuk enkripsi TDE.

- Timezone— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung.

- VpcSecurityGroups – Susunan objek VpcSecurityGroupMembership.

  Menyediakan daftar elemen grup keamanan VPC tempat instans DB berada.

Kesalahan

- DBInstanceAlreadyExistsFault
- Tidak cukup DBInstance CapacityFault
- DBParameterGroupNotFoundFault
- DBSecurityGroupNotFoundFault
- InstanceQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- Tidak valid DBCluster StateFault
- InvalidSubnet
- Tidak valid VPCNetwork StateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBClusterNotFoundFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- KMSKeyNotAccessibleFault
- DomainNotFoundFault

Hapus DBInstance (tindakan)

Nama AWS CLI untuk API ini adalah:. `delete-db-instance`

DBInstance Tindakan Hapus menghapus instans DB yang telah disediakan sebelumnya. Ketika Anda menghapus sebuah klaster, semua backup otomatis untuk instans tersebut dihapus dan tidak dapat dipulihkan. Snapshot DB manual dari instans DB yang akan dihapus oleh `DeleteDBInstance` tidak dihapus.

Jika Anda meminta snapshot DB akhir status instans DB Amazon Neptune misalnya adalah `deleting` sampai snapshot DB dibuat. Tindakan API `DescribeDBInstance` digunakan untuk memantau status operasi ini. Tindakan tidak dapat dibatalkan atau dikembalikan setelah dikirimkan.

Perhatikan bahwa ketika instans DB dalam keadaan kegagalan dan memiliki status `failed`, `incompatible-restore`, atau `incompatible-network`, Anda hanya dapat menghapusnya ketika parameter `SkipFinalSnapshot` diatur ke `true`.

Anda tidak dapat menghapus instans DB jika itu adalah satu-satunya instans dalam klaster DB, atau jika memiliki perlindungan penghapusan diaktifkan.

Permintaan

- DBInstanceIdentifier(dalam CLI:`--db-instance-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi instans DB untuk instans DB yang akan dihapus. Parameter ini tidak peka huruf besar-kecil.

  Kendala:

  - Harus cocok dengan nama instans DB yang ada.
- FinalDBSnapshotIdentifier(dalam CLI:`--final-db-snapshot-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  DBSnapshotIdentifier dari yang baru DBSnapshot dibuat saat SkipFinalSnapshot diatur ke`false`.

  > **Note**
  >
  > Menentukan parameter ini dan juga mengatur SkipFinalShapshot parameter ke hasil yang benar dalam kesalahan.

Batasan:

- Harus huruf atau angka berisi 1 sampai 255 karakter.
- Karakter pertamanya harus berupa huruf
- Tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung yang berurutan
- Tidak dapat ditentukan saat menghapus Replika Baca.

- SkipFinalSnapshot(dalam CLI:`--skip-final-snapshot`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah snapshot DB akhir dibuat sebelum instans DB dihapus. Jika `true` ditentukan, tidak DBSnapshot ada yang dibuat. Jika `false` ditentukan, snapshot DB dibuat sebelum instans DB dihapus.

  Perhatikan bahwa ketika instans DB berada dalam status kegagalan dan memiliki status 'gagal', 'incompatible-restore', atau 'incompatible-network ', itu hanya dapat dihapus ketika parameter disetel ke SkipFinalSnapshot “true”.

  Tentukan `true` saat menghapus Replika Baca.

  > **Note**
  >
  > Parameter DBSnapshot Pengenal Akhir harus ditentukan jika SkipFinalSnapshot ada`false`.

  Default: `false`

Respons

Berisi detail dari instans DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBInstances”.

- AutoMinorVersionUpgrade— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan bahwa patch versi minor diterapkan secara otomatis.

- AvailabilityZone— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama Availability Zone tempat instans DB tersebut berada.
- BackupRetentionPeriod— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.
- CACertificateIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi sertifikat CA untuk instans DB ini.
- CopyTagsToSnapshot— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah tanda disalin dari instans DB ke snapshot instans DB.
- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika instans DB adalah anggota klaster DB, berisi nama klaster DB tempat instans DB tersebut menjadi anggota.
- DBInstanceArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk instans DB.
- DBInstanceClass— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama kelas kapasitas komputasi dan memori instans DB.
- DBInstanceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi basis data yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi sebuah instans DB.
- DbInstancePort— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port tempat instans DB mendengarkan. Jika instans DB adalah bagian dari klaster DB, ini bisa menjadi port yang berbeda dari port klaster DB.
- DBInstanceStatus— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari basis data ini.
- DbiResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi Wilayah Amazon unik yang tetap untuk instans DB. Pengidentifikasi ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk instans DB diakses.
- DBName— String, tipe: `string` (string yang dikodekan UTF-8).

Nama database.

- DBParameterGroups – Susunan objek DBParameterGroupStatus.

  Menyediakan daftar grup parameter DB yang diterapkan ke instans DB ini.

- DBSecurityGroups – Susunan objek DBSecurityGroupMembership.

  Menyediakan Daftar elemen grup keamanan DB yang hanya berisi subelemen `DBSecurityGroup.Name` dan `DBSecurityGroup.Status`.

- DBSubnetGroup — Sebuah objek DBSubnetKelompok.

  Menentukan informasi tentang grup subnet yang terkait dengan instans DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Instans tidak dapat dihapus saat perlindungan penghapusan diaktifkan. Lihat Menghapus instans DB.

- DomainMemberships – Susunan objek DomainMembership.

  Tidak didukung

- EnabledCloudwatchLogsExports— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang instans DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log.

- Endpoint — Sebuah objek Titik Akhir.

  Menentukan titik akhir koneksi.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk instans DB ini.

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.

- EnhancedMonitoringResourceArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) dari aliran CloudWatch log Amazon Logs yang menerima data metrik Pemantauan yang Ditingkatkan untuk instans DB.

- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

True jika autentikasi Amazon Identity and Access Management (IAM) diaktifkan, dan jika sebaliknya akan false.

- InstanceCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menyediakan tanggal dan waktu saat instans DB dibuat.

- Iops— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan nilai Provisioned IOPS (operasi I/O per detik).

- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung: Enkripsi untuk instans DB dikelola oleh klaster DB.

- LatestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.

- LicenseModel— String, tipe: `string` (string yang dikodekan UTF-8).

  Informasi model lisensi untuk instans DB ini.

- MonitoringInterval— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Interval, dalam detik, antara titik-titik saat metrik Pemantauan yang Ditingkatkan dikumpulkan untuk instans DB.

- MonitoringRoleArn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN untuk peran IAM yang memungkinkan Neptunus mengirim metrik Pemantauan yang Ditingkatkan ke Log Amazon. CloudWatch

- MultiAZ— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan jika instans DB adalah deployment Multi-AZ.

- PendingModifiedValues — Sebuah objek PendingModifiedValues.

  Menentukan bahwa perubahan pada instans DB sedang tertunda. Elemen ini disertakan hanya ketika perubahan tertunda. Perubahan spesifik diidentifikasi oleh sub elemen.

- PreferredBackupWindow— String, tipe: `string` (string yang dikodekan UTF-8).

Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.

- PreferredMaintenanceWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).

- PromotionTier— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Nilai yang menentukan urutan di mana Replika Baca dipromosikan ke instans primer setelah kegagalan instans primer yang ada.

- PubliclyAccessible— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Bendera ini seharusnya tidak lagi digunakan.

- ReadReplicaDBClusterIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi klaster DB yang merupakan Replika Baca instans DB ini.

- ReadReplicaDBInstanceIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu pengidentifikasi atau lebih dari Replika Baca yang terkait dengan instans DB ini.

- ReadReplicaSourceDBInstanceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi instans DB sumber jika instans DB ini adalah Replika Baca.

- SecondaryAvailabilityZone— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika ada, menentukan nama Availability Zone sekunder untuk instans DB dengan dukungan multi-AZ.

- StatusInfos – Susunan objek DBInstanceStatusInfo.

  Status Replika Baca. Jika instans bukan Replika Baca, ini kosong.

- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Tidak didukung: Enkripsi untuk instans DB dikelola oleh klaster DB.

- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan jenis penyimpanan yang terkait dengan instans DB.

- TdeCredentialArn— String, tipe: `string` (string yang dikodekan UTF-8).

ARN dari penyimpanan kunci dengan yang terkait dengan instans untuk enkripsi TDE.

- Timezone— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung.

- VpcSecurityGroups – Susunan objek VpcSecurityGroupMembership.

  Menyediakan daftar elemen grup keamanan VPC tempat instans DB berada.

Kesalahan

- DBInstanceNotFoundFault
- Tidak valid DBInstance StateFault
- DBSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault
- Tidak valid DBCluster StateFault

Memodifikasi DBInstance (tindakan)

Nama AWS CLI untuk API ini adalah:. `modify-db-instance`

Mengubah pengaturan untuk instans DB. Anda dapat mengubah satu atau lebih parameter konfigurasi basis data dengan menentukan parameter ini dan nilai-nilai baru dalam permintaan. Untuk mempelajari modifikasi apa yang dapat Anda buat untuk instans DB Anda, panggil the section called “DescribeValidDBInstanceModifikasi” sebelum Anda memanggil the section called “Memodifikasi DBInstance”.

Permintaan

- AllowMajorVersionUpgrade(dalam CLI:`--allow-major-version-upgrade`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan bahwa peningkatan versi utama diizinkan. Mengubah parameter ini tidak mengakibatkan pemadaman dan perubahan diterapkan secara asinkron sesegera mungkin.

- ApplyImmediately(dalam CLI:`--apply-immediately`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

Menentukan apakah modifikasi dalam permintaan ini dan modifikasi yang tertunda diterapkan secara asinkron sesegera mungkin, terlepas dari pengaturan `PreferredMaintenanceWindow` untuk instans DB.

Jika parameter ini diatur ke `false`, perubahan ke instans DB diterapkan selama jendela pemeliharaan berikutnya. Beberapa perubahan parameter dapat menyebabkan pemadaman dan diterapkan pada panggilan berikutnya ke the section called “Reboot DBInstance”, atau boot ulang kegagalan berikutnya.

Default: `false`

- AutoMinorVersionUpgrade(dalam CLI:`--auto-minor-version-upgrade`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang menunjukkan apakah peningkatan versi kecil diterapkan secara otomatis ke instans DB selama jendela pemeliharaan. Mengubah parameter ini tidak mengakibatkan pemadaman kecuali dalam kasus berikut, dan perubahan diterapkan secara asinkron sesegera mungkin. Pemadaman akan mengakibatkan jika parameter ini diatur ke `true` selama jendela pemeliharaan, dan versi minor yang lebih baru tersedia, dan Neptune telah mengaktifkan patching otomatis untuk versi mesin tersebut.
- BackupRetentionPeriod(dalam CLI:`--backup-retention-period`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Tidak berlaku. Periode retensi untuk backup otomatis dikelola oleh klaster DB. Untuk informasi selengkapnya, lihat the section called “Memodifikasi DBCluster”.

  Default: Menggunakan pengaturan yang ada
- CACertificateIdentifier(dalam CLI:`--ca-certificate-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan sertifikat yang perlu dikaitkan dengan instans.
- CloudwatchLogsExportConfiguration(dalam CLI:`--cloudwatch-logs-export-configuration`) — Sebuah CloudwatchLogsExportConfiguration objek.

  Pengaturan konfigurasi untuk jenis log yang akan diaktifkan untuk ekspor ke CloudWatch Log untuk instans DB atau cluster DB tertentu.
- CopyTagsToSnapshot(dalam CLI:`--copy-tags-to-snapshot`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

True untuk menyalin semua tag dari instans DB untuk snapshot instans DB, dan sebaliknya false. Default-nya adalah salah.

- DBInstanceClass(dalam CLI:`--db-instance-class`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Kapasitas komputasi dan memori baru instans DB; misalnya, `db.m4.large`. Tidak semua kelas instans DB tersedia di semua Wilayah Amazon.

  Jika Anda memodifikasi kelas instans DB, pemadaman terjadi selama perubahan. Perubahan diterapkan selama jendela pemeliharaan berikutnya, kecuali `ApplyImmediately` ditentukan sebagai `true` untuk permintaan ini.

  Default: Menggunakan pengaturan yang ada

- DBInstanceIdentifier(dalam CLI:`--db-instance-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi instans DB. Nilai ini disimpan sebagai string huruf kecil.

  Batasan:

  - Harus cocok dengan pengenal yang sudah ada DBInstance.

- DBParameterGroupName(dalam CLI:`--db-parameter-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter DB untuk diterapkan ke instans DB. Mengubah pengaturan ini tidak mengakibatkan pemadaman listrik. Nama grup parameter itu sendiri segera berubah, tetapi perubahan parameter yang sebenarnya tidak diterapkan sampai Anda me-reboot instans tanpa failover. Instans db TIDAK akan reboot secara otomatis dan perubahan parameter TIDAK akan diterapkan selama jendela pemeliharaan berikutnya.

  Default: Menggunakan pengaturan yang ada

  Kendala: Grup parameter DB harus dalam keluarga grup parameter DB yang sama seperti instans DB ini.

- DBPortNumber(dalam CLI:`--db-port-number`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Nomor port tempat basis data menerima koneksi.

Nilai parameter `DBPortNumber` tidak boleh cocok dengan nilai port mana pun yang ditentukan untuk opsi dalam grup opsi untuk instans DB.

Basis data Anda akan restart ketika Anda mengubah nilai `DBPortNumber` terlepas dari nilai parameter `ApplyImmediately`.

Default: `8182`

- DBSecurityGroups(dalam CLI:`--db-security-groups`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar grup keamanan DB untuk mengotorisasi di instans DB ini. Mengubah pengaturan ini tidak mengakibatkan pemadaman dan perubahan diterapkan secara asinkron sesegera mungkin.

  Batasan:

  - Jika disediakan, harus cocok dengan DBSecurity Grup yang ada.
- DBSubnetGroupName(dalam CLI:`--db-subnet-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Grup subnet DB baru untuk instans DB. Anda dapat menggunakan parameter ini untuk memindahkan instans DB Anda ke VPC yang berbeda.

  Mengubah grup subnet menyebabkan pemadaman selama perubahan. Kecuali parameter `ApplyImmediately` diatur ke `true`, perubahan akan diterapkan selama jendela pemeliharaan berikutnya.

  Kendala: Jika disediakan, harus cocok dengan nama Grup yang ada. DBSubnet

  Contoh: `mySubnetGroup`
- DeletionProtection(dalam CLI:`--deletion-protection`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang menunjukkan apakah instans DB mengaktifkan perlindungan penghapusan. Basis data tidak dapat dihapus saat perlindungan penghapusan diaktifkan. Secara default, perlindungan penghapusan dinonaktifkan. Lihat Menghapus instans DB.
- Domain(dalam CLI:`--domain`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung.

- DomainIAMRoleName(dalam CLI:`--domain-iam-role-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung

- EnableIAMDatabaseAuthentication(dalam CLI:`--enable-iam-database-authentication`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  True untuk mengaktifkan pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data, dan sebaliknya false.

  Anda dapat mengaktifkan autentikasi basis data IAM untuk mesin basis data berikut

  Tidak berlaku. Memetakan akun IAM Amazon ke akun basis data dikelola oleh klaster DB. Untuk informasi selengkapnya, lihat the section called “Memodifikasi DBCluster”.

  Default: `false`

- EngineVersion(dalam CLI:`--engine-version`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nomor versi mesin basis data yang akan ditingkatkan. Saat ini, pengaturan parameter ini tidak berpengaruh. Untuk meng-upgrade mesin basis data Anda untuk rilis terbaru, gunakan API the section called “ApplyPendingMaintenanceAction”.

- Iops(dalam CLI:`--iops`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Nilai Provisioned IOPS (operasi I/O per detik) yang baru untuk instans.

  Mengubah pengaturan ini tidak menyebabkan pemadaman dan perubahan diterapkan selama jendela pemeliharaan berikutnya kecuali parameter `ApplyImmediately` diatur ke `true` untuk permintaan ini.

  Default: Menggunakan pengaturan yang ada

- MonitoringInterval(dalam CLI:`--monitoring-interval`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Interval, dalam detik, antara titik-titik saat metrik Pemantauan yang Ditingkatkan dikumpulkan untuk instans DB. Untuk menonaktifkan pengumpulan metrik Pemantauan yang Ditingkatkan, tentukan 0. Default-nya adalah 0.

Jika `MonitoringRoleArn` ditentukan, maka Anda juga harus mengatur `MonitoringInterval` ke nilai selain 0.

Nilai Valid: `0, 1, 5, 10, 15, 30, 60`

- MonitoringRoleArn(dalam CLI:`--monitoring-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN untuk peran IAM yang memungkinkan Neptunus mengirim metrik pemantauan yang ditingkatkan ke Amazon Logs. CloudWatch Misalnya, `arn:aws:iam:123456789012:role/emaccess`.

  Jika `MonitoringInterval` diatur ke nilai selain 0, maka Anda harus menyediakan nilai `MonitoringRoleArn`.
- MultiAZ(dalam CLI:`--multi-az`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan jika instans DB adalah deployment Multi-AZ. Mengubah parameter ini tidak menyebabkan pemadaman dan perubahan diterapkan selama jendela pemeliharaan berikutnya kecuali parameter `ApplyImmediately` diatur ke `true` untuk permintaan ini.
- NewDBInstanceIdentifier(dalam CLI:`--new-db-instance-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi instans DB baru untuk instans DB ketika mengubah nama instans DB. Ketika Anda mengubah pengidentifikasi instans DB, reboot instans akan terjadi segera jika Anda mengatur `Apply Immediately` ke true, atau akan terjadi selama jendela pemeliharaan berikutnya jika `Apply Immediately` diatur ke false. Nilai ini disimpan sebagai string huruf kecil.

  Kendala:

  - Harus berisi 1 hingga 63 huruf, angka, atau tanda hubung.
  - Karakter pertama harus berupa huruf.
  - Tidak dapat diakhiri dengan tanda hubung atau mengandung dua tanda hubung berturut-turut.

  Contoh: `mydbinstance`
- PreferredBackupWindow(dalam CLI:`--preferred-backup-window`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Rentang waktu harian selama pencadangan otomatis dibuat jika pencadangan otomatis diaktifkan.

Tidak berlaku. Kisaran waktu harian untuk membuat backup otomatis dikelola oleh klaster DB. Untuk informasi selengkapnya, lihat the section called “Memodifikasi DBCluster”.

Kendala:

- Harus dalam format hh24:mi-hh24:mi
- Harus dalam Waktu Universal Terkoordinasi (UTC).
- Tidak boleh bertentangan dengan pemeliharaan jendela yang diinginkan
- Harus setidaknya 30 menit

- PreferredMaintenanceWindow(dalam CLI:`--preferred-maintenance-window`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Rentang waktu mingguan (dalam UTC) untuk melakukan pemeliharaan sistem dapat terjadi, yang dapat mengakibatkan pemadaman. Mengubah parameter ini tidak mengakibatkan pemadaman kecuali dalam situasi berikut, dan perubahan diterapkan secara asinkron sesegera mungkin. Jika ada tindakan tertunda yang menyebabkan boot ulang, dan jendela pemeliharaan diubah untuk menyertakan waktu saat ini, lalu mengubah parameter ini akan menyebabkan boot ulang instans DB. Jika memindahkan jendela ke waktu saat ini, harus ada minimal 30 menit antara waktu saat ini dan waktu selesai jendela untuk memastikan setiap perubahan yang tertunda sudah diterapkan.

  Default: Menggunakan pengaturan yang ada

  Format: ddd:hh24:mi-ddd:hh24:mi

  Hari yang Valid: Sen | Sel | Rab | Kam | Jum | Sab | Min

  Kendala: Harus setidaknya 30 menit.

- PromotionTier(dalam CLI:`--promotion-tier`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Nilai yang menentukan urutan di mana Replika Baca dipromosikan ke instans primer setelah kegagalan instans primer yang ada.

  Default: 1

  Nilai yang Valid: 0-15

- PubliclyAccessible(dalam CLI:`--publicly-accessible`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

Bendera ini seharusnya tidak lagi digunakan.

- StorageType(dalam CLI:`--storage-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak berlaku. Di Neptunus jenis penyimpanan dikelola pada tingkat DB Cluster.

- TdeCredentialArn(dalam CLI:`--tde-credential-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari penyimpanan kunci dengan yang terkait dengan instans untuk enkripsi TDE.

- TdeCredentialPassword(dalam CLI:`--tde-credential-password`) — a SensitiveString, tipe: `string` (string yang dikodekan UTF-8).

  Kata sandi untuk ARN yang diberikan dari penyimpanan kunci untuk mengakses perangkat.

- VpcSecurityGroupIds(dalam CLI:`--vpc-security-group-ids`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar grup keamanan EC2 VPC untuk mengotorisasi instans DB ini. Perubahan ini diterapkan secara asinkron, sesegera mungkin.

  Tidak berlaku. Daftar terkait grup keamanan EC2 VPC dikelola oleh cluster DB. Untuk informasi selengkapnya, lihat the section called “Memodifikasi DBCluster”.

  Batasan:

  - Jika disediakan, harus cocok yang ada VpcSecurityGroupIds.

Respons

Berisi detail dari instans DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBInstances”.

- AutoMinorVersionUpgrade— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan bahwa patch versi minor diterapkan secara otomatis.

- AvailabilityZone— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama Availability Zone tempat instans DB tersebut berada.

- BackupRetentionPeriod— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.
- CACertificateIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi sertifikat CA untuk instans DB ini.
- CopyTagsToSnapshot— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah tanda disalin dari instans DB ke snapshot instans DB.
- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika instans DB adalah anggota klaster DB, berisi nama klaster DB tempat instans DB tersebut menjadi anggota.
- DBInstanceArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk instans DB.
- DBInstanceClass— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama kelas kapasitas komputasi dan memori instans DB.
- DBInstanceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi basis data yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi sebuah instans DB.
- DbInstancePort— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port tempat instans DB mendengarkan. Jika instans DB adalah bagian dari klaster DB, ini bisa menjadi port yang berbeda dari port klaster DB.
- DBInstanceStatus— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari basis data ini.
- DbiResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi Wilayah Amazon unik yang tetap untuk instans DB. Pengidentifikasi ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk instans DB diakses.
- DBName— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama database.
- DBParameterGroups – Susunan objek DBParameterGroupStatus.

Menyediakan daftar grup parameter DB yang diterapkan ke instans DB ini.

- DBSecurityGroups – Susunan objek DBSecurityGroupMembership.

  Menyediakan Daftar elemen grup keamanan DB yang hanya berisi subelemen `DBSecurityGroup.Name` dan `DBSecurityGroup.Status`.

- DBSubnetGroup — Sebuah objek DBSubnetKelompok.

  Menentukan informasi tentang grup subnet yang terkait dengan instans DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Instans tidak dapat dihapus saat perlindungan penghapusan diaktifkan. Lihat Menghapus instans DB.

- DomainMemberships – Susunan objek DomainMembership.

  Tidak didukung

- EnabledCloudwatchLogsExports— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang instans DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log.

- Endpoint — Sebuah objek Titik Akhir.

  Menentukan titik akhir koneksi.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk instans DB ini.

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.

- EnhancedMonitoringResourceArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) dari aliran CloudWatch log Amazon Logs yang menerima data metrik Pemantauan yang Ditingkatkan untuk instans DB.

- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  True jika autentikasi Amazon Identity and Access Management (IAM) diaktifkan, dan jika sebaliknya akan false.

- InstanceCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menyediakan tanggal dan waktu saat instans DB dibuat.

- Iops— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan nilai Provisioned IOPS (operasi I/O per detik).

- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung: Enkripsi untuk instans DB dikelola oleh klaster DB.

- LatestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.

- LicenseModel— String, tipe: `string` (string yang dikodekan UTF-8).

  Informasi model lisensi untuk instans DB ini.

- MonitoringInterval— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Interval, dalam detik, antara titik-titik saat metrik Pemantauan yang Ditingkatkan dikumpulkan untuk instans DB.

- MonitoringRoleArn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN untuk peran IAM yang memungkinkan Neptunus mengirim metrik Pemantauan yang Ditingkatkan ke Log Amazon. CloudWatch

- MultiAZ— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan jika instans DB adalah deployment Multi-AZ.

- PendingModifiedValues — Sebuah objek PendingModifiedValues.

  Menentukan bahwa perubahan pada instans DB sedang tertunda. Elemen ini disertakan hanya ketika perubahan tertunda. Perubahan spesifik diidentifikasi oleh sub elemen.

- PreferredBackupWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.

- PreferredMaintenanceWindow— String, tipe: `string` (string yang dikodekan UTF-8).

Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).

- PromotionTier— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Nilai yang menentukan urutan di mana Replika Baca dipromosikan ke instans primer setelah kegagalan instans primer yang ada.

- PubliclyAccessible— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Bendera ini seharusnya tidak lagi digunakan.

- ReadReplicaDBClusterIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi klaster DB yang merupakan Replika Baca instans DB ini.

- ReadReplicaDBInstanceIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu pengidentifikasi atau lebih dari Replika Baca yang terkait dengan instans DB ini.

- ReadReplicaSourceDBInstanceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi instans DB sumber jika instans DB ini adalah Replika Baca.

- SecondaryAvailabilityZone— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika ada, menentukan nama Availability Zone sekunder untuk instans DB dengan dukungan multi-AZ.

- StatusInfos – Susunan objek DBInstanceStatusInfo.

  Status Replika Baca. Jika instans bukan Replika Baca, ini kosong.

- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Tidak didukung: Enkripsi untuk instans DB dikelola oleh klaster DB.

- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan jenis penyimpanan yang terkait dengan instans DB.

- TdeCredentialArn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari penyimpanan kunci dengan yang terkait dengan instans untuk enkripsi TDE.

- Timezone— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung.

- VpcSecurityGroups – Susunan objek VpcSecurityGroupMembership.

  Menyediakan daftar elemen grup keamanan VPC tempat instans DB berada.

Kesalahan

- Tidak valid DBInstance StateFault
- Tidak valid DBSecurity GroupStateFault
- DBInstanceAlreadyExistsFault
- DBInstanceNotFoundFault
- DBSecurityGroupNotFoundFault
- DBParameterGroupNotFoundFault
- Tidak cukup DBInstance CapacityFault
- StorageQuotaExceededFault
- Tidak valid VPCNetwork StateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBUpgradeDependencyFailureFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- CertificateNotFoundFault
- DomainNotFoundFault

Reboot DBInstance (tindakan)

Nama AWS CLI untuk API ini adalah:. `reboot-db-instance`

Anda mungkin perlu mem-boot ulang instans DB Anda, biasanya untuk alasan pemeliharaan. Misalnya, jika Anda melakukan modifikasi tertentu, atau jika Anda mengubah grup parameter DB yang terkait dengan instans DB, Anda harus mem-boot ulang instans perubahan yang akan diterapkan.

Boot ulang instans DB akan mengaktifkan ulang layanan mesin basis data. Boot ulang instans DB akan menyebabkan pemadaman sementara, dan selama waktu ini, status instans DB diatur ke mem-boot ulang.

Permintaan

- DBInstanceIdentifier(dalam CLI:`--db-instance-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi instans DB. Parameter ini disimpan sebagai string huruf kecil.

  Batas:

  - Harus cocok dengan pengenal yang sudah ada DBInstance.
- ForceFailover(dalam CLI:`--force-failover`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Saat `true`, boot ulang dilakukan melalui failover MultiAZ.

  Kendala: Anda tidak dapat menentukan `true` jika instans tidak dikonfigurasi untuk MultiAZ.

Respons

Berisi detail dari instans DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBInstances”.

- AutoMinorVersionUpgrade— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan bahwa patch versi minor diterapkan secara otomatis.
- AvailabilityZone— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama Availability Zone tempat instans DB tersebut berada.
- BackupRetentionPeriod— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.
- CACertificateIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi sertifikat CA untuk instans DB ini.
- CopyTagsToSnapshot— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah tanda disalin dari instans DB ke snapshot instans DB.
- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

Jika instans DB adalah anggota klaster DB, berisi nama klaster DB tempat instans DB tersebut menjadi anggota.

- DBInstanceArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk instans DB.

- DBInstanceClass— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama kelas kapasitas komputasi dan memori instans DB.

- DBInstanceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi basis data yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi sebuah instans DB.

- DbInstancePort— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port tempat instans DB mendengarkan. Jika instans DB adalah bagian dari klaster DB, ini bisa menjadi port yang berbeda dari port klaster DB.

- DBInstanceStatus— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari basis data ini.

- DbiResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi Wilayah Amazon unik yang tetap untuk instans DB. Pengenal ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk instans DB diakses.

- DBName— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama database.

- DBParameterGroups – Susunan objek DBParameterGroupStatus.

  Menyediakan daftar grup parameter DB yang diterapkan ke instans DB ini.

- DBSecurityGroups – Susunan objek DBSecurityGroupMembership.

  Menyediakan Daftar elemen grup keamanan DB yang hanya berisi subelemen `DBSecurityGroup.Name` dan `DBSecurityGroup.Status`.

- DBSubnetGroup — Sebuah objek DBSubnetKelompok.

  Menentukan informasi tentang grup subnet yang terkait dengan instans DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Instans tidak dapat dihapus saat perlindungan penghapusan diaktifkan. Lihat Menghapus instans DB.

- DomainMemberships – Susunan objek DomainMembership.

  Tidak didukung

- EnabledCloudwatchLogsExports— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang instans DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log.

- Endpoint — Sebuah objek Titik Akhir.

  Menentukan titik akhir koneksi.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk instans DB ini.

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.

- EnhancedMonitoringResourceArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) dari aliran CloudWatch log Amazon Logs yang menerima data metrik Pemantauan yang Ditingkatkan untuk instans DB.

- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  True jika autentikasi Amazon Identity and Access Management (IAM) diaktifkan, dan jika sebaliknya akan false.

- InstanceCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menyediakan tanggal dan waktu saat instans DB dibuat.

- Iops— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan nilai Provisioned IOPS (operasi I/O per detik).

- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung: Enkripsi untuk instans DB dikelola oleh klaster DB.

- LatestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.
- LicenseModel— String, tipe: `string` (string yang dikodekan UTF-8).

  Informasi model lisensi untuk instans DB ini.
- MonitoringInterval— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Interval, dalam detik, antara titik-titik saat metrik Pemantauan yang Ditingkatkan dikumpulkan untuk instans DB.
- MonitoringRoleArn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN untuk peran IAM yang memungkinkan Neptunus mengirim metrik Pemantauan yang Ditingkatkan ke Log Amazon. CloudWatch
- MultiAZ— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan jika instans DB adalah deployment Multi-AZ.
- PendingModifiedValues — Sebuah objek PendingModifiedValues.

  Menentukan bahwa perubahan pada instans DB sedang tertunda. Elemen ini disertakan hanya ketika perubahan tertunda. Perubahan spesifik diidentifikasi oleh sub elemen.
- PreferredBackupWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.
- PreferredMaintenanceWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).
- PromotionTier— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Nilai yang menentukan urutan di mana Replika Baca dipromosikan ke instans primer setelah kegagalan instans primer yang ada.
- PubliclyAccessible— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Bendera ini seharusnya tidak lagi digunakan.
- ReadReplicaDBClusterIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

Berisi satu atau lebih pengidentifikasi klaster DB yang merupakan Replika Baca instans DB ini.

- ReadReplicaDBInstanceIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu pengidentifikasi atau lebih dari Replika Baca yang terkait dengan instans DB ini.

- ReadReplicaSourceDBInstanceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi instans DB sumber jika instans DB ini adalah Replika Baca.

- SecondaryAvailabilityZone— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika ada, menentukan nama Availability Zone sekunder untuk instans DB dengan dukungan multi-AZ.

- StatusInfos – Susunan objek DBInstanceStatusInfo.

  Status Replika Baca. Jika instans bukan Replika Baca, ini kosong.

- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Tidak didukung: Enkripsi untuk instans DB dikelola oleh klaster DB.

- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan jenis penyimpanan yang terkait dengan instans DB.

- TdeCredentialArn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari penyimpanan kunci dengan yang terkait dengan instans untuk enkripsi TDE.

- Timezone— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung.

- VpcSecurityGroups – Susunan objek VpcSecurityGroupMembership.

  Menyediakan daftar elemen grup keamanan VPC tempat instans DB berada.

Kesalahan

- Tidak valid DBInstance StateFault
- DBInstanceNotFoundFault

Jelaskan DBInstances (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-db-instances`

Mengembalikan informasi tentang instans yang disediakan, dan mendukung pemberian nomor halaman.

> **Note**
>
> Operasi ini juga dapat mengembalikan informasi untuk instans Amazon RDS dan instans Amazon DocDB.

Permintaan

- DBInstanceIdentifier(dalam CLI:`--db-instance-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengenal instans yang disediakan pengguna. Jika parameter ini ditentukan, informasi hanya dari instans DB tertentu yang dikembalikan. Parameter ini tidak peka huruf besar-kecil.

  Batasan:

  - Jika disediakan, harus cocok dengan pengenal yang sudah ada DBInstance.
- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Filter yang menentukan satu atau lebih instans DB untuk dideskripsikan.

  Filter yang didukung:

  - `db-cluster-id`- Menerima pengidentifikasi cluster DB dan DB cluster Amazon Resource Names ()ARNs. Daftar hasil hanya akan menyertakan informasi tentang instans DB yang terkait dengan cluster DB yang diidentifikasi oleh ini. ARNs
  - `engine` - Menerima nama mesin (seperti `neptune`), dan membatasi daftar hasil untuk instans DB yang dibuat oleh mesin tersebut.

  Misalnya, untuk memanggil API ini dari Amazon CLI dan memfilter sehingga hanya instans DB Neptune yang dikembalikan, Anda bisa menggunakan perintah berikut:

Example

```
aws neptune describe-db-instances \
            --filters  Name=engine,Values=neptune
```

- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan `DescribeDBInstances` sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

  Default: 100

  Kendala: Minimal 20, maksimum 100.

Respons

- DBInstances – Susunan objek DBInstance.

  Daftar instans the section called “DBInstance”.

- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan sebelumnya. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

Kesalahan

- DBInstanceNotFoundFault

DescribeOrderableDBInstanceOpsi (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-orderable-db-instance-options`

Mengembalikan daftar opsi instans DB yang dapat diurutkan untuk mesin yang ditentukan.

Permintaan

- DBInstanceClass(dalam CLI:`--db-instance-class`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nilai filter kelas instans DB. Tentukan parameter ini untuk hanya menampilkan penawaran yang tersedia yang cocok dengan kelas instans DB yang ditentukan.
- Engine(dalam CLI:`--engine`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama dari mesin untuk mengambil opsi instans DB.
- EngineVersion(dalam CLI:`--engine-version`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nilai filter versi mesin. Tentukan parameter ini untuk hanya menampilkan penawaran yang tersedia yang cocok dengan versi mesin yang ditentukan.
- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Parameter ini saat ini tidak didukung.
- LicenseModel(dalam CLI:`--license-model`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nilai filter model lisensi. Tentukan parameter ini untuk hanya menampilkan penawaran yang tersedia yang cocok dengan model lisensi yang ditentukan.
- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan DescribeOrderable DBInstance Opsi sebelumnya. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.
- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

Default: 100

Kendala: Minimal 20, maksimum 100.

- Vpc(dalam CLI:`--vpc`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai filter VPC. Tentukan parameter ini untuk hanya menampilkan penawaran VPC atau non-VPC yang tersedia.

Respons

- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan Opsi yang Dapat Dipesan sebelumnyaDBInstance. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.
- OrderableDBInstanceOptions – Susunan objek Opsi yang Dapat Dipesan DBInstance.

  Struktur the section called “Opsi yang Dapat Dipesan DBInstance” yang berisi informasi tentang opsi yang dapat dipesan untuk instans DB.

DescribeValidDBInstanceModifikasi (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-valid-db-instance-modifications`

Anda dapat memanggil the section called “DescribeValidDBInstanceModifikasi” untuk mempelajari modifikasi apa yang dapat Anda buat untuk instans DB Anda. Anda dapat menggunakan informasi ini saat Anda memanggil the section called “Memodifikasi DBInstance”.

Permintaan

- DBInstanceIdentifier(dalam CLI:`--db-instance-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

Pengidentifikasi pelanggan atau ARN instans DB Anda.

Respons

Informasi tentang modifikasi valid yang dapat Anda buat untuk instans DB Anda. Berisi hasil dari panggilan sukses ke tindakan the section called “DescribeValidDBInstanceModifikasi”. Anda dapat menggunakan informasi ini saat Anda memanggil the section called “Memodifikasi DBInstance”.

- Storage – Susunan objek ValidStorageOptions.

  Opsi penyimpanan yang valid untuk instans DB Anda.

Kesalahan

- DBInstanceNotFoundFault
- Tidak valid DBInstance StateFault

Struktur:

DBInstance (struktur)

Berisi detail dari instans DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBInstances”.

Bidang

- AutoMinorVersionUpgrade— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan bahwa patch versi minor diterapkan secara otomatis.
- AvailabilityZone— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama Availability Zone tempat instans DB tersebut berada.
- BackupRetentionPeriod— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.

- CACertificateIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi sertifikat CA untuk instans DB ini.
- CopyTagsToSnapshot— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah tanda disalin dari instans DB ke snapshot instans DB.
- DBClusterIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jika instans DB adalah anggota klaster DB, berisi nama klaster DB tempat instans DB tersebut menjadi anggota.
- DBInstanceArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk instans DB.
- DBInstanceClass— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama kelas kapasitas komputasi dan memori instans DB.
- DBInstanceIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi basis data yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi sebuah instans DB.
- DbInstancePort— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port tempat instans DB mendengarkan. Jika instans DB adalah bagian dari klaster DB, ini bisa menjadi port yang berbeda dari port klaster DB.
- DBInstanceStatus— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari basis data ini.
- DbiResourceId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi Wilayah Amazon unik yang tetap untuk instans DB. Pengenal ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk instans DB diakses.
- DBName— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama database.
- DBParameterGroups— Ini adalah Array DBParameterGroupStatus objek.

  Menyediakan daftar grup parameter DB yang diterapkan ke instans DB ini.
- DBSecurityGroups— Ini adalah Array DBSecurityGroupMembership objek.

Menyediakan Daftar elemen grup keamanan DB yang hanya berisi subelemen `DBSecurityGroup.Name` dan `DBSecurityGroup.Status`.

- DBSubnetGroupIni adalah sebuah DBSubnetKelompok objek.

  Menentukan informasi tentang grup subnet yang terkait dengan instans DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.
- DeletionProtection— Ini adalah BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Instans tidak dapat dihapus saat perlindungan penghapusan diaktifkan. Lihat Menghapus instans DB.
- DomainMemberships— Ini adalah Array DomainMembership objek.

  Tidak didukung
- EnabledCloudwatchLogsExports— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang instans DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log.
- EndpointIni adalah sebuah Titik Akhir objek.

  Menentukan titik akhir koneksi.
- Engine— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk instans DB ini.
- EngineVersion— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.
- EnhancedMonitoringResourceArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) dari aliran CloudWatch log Amazon Logs yang menerima data metrik Pemantauan yang Ditingkatkan untuk instans DB.
- IAMDatabaseAuthenticationEnabled— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  True jika autentikasi Amazon Identity and Access Management (IAM) diaktifkan, dan jika sebaliknya akan false.
- InstanceCreateTime— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

Menyediakan tanggal dan waktu saat instans DB dibuat.

- Iops— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan nilai Provisioned IOPS (operasi I/O per detik).

- KmsKeyId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung: Enkripsi untuk instans DB dikelola oleh klaster DB.

- LatestRestorableTime— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.

- LicenseModel— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Informasi model lisensi untuk instans DB ini.

- MonitoringInterval— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Interval, dalam detik, antara titik-titik saat metrik Pemantauan yang Ditingkatkan dikumpulkan untuk instans DB.

- MonitoringRoleArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  ARN untuk peran IAM yang memungkinkan Neptunus mengirim metrik Pemantauan yang Ditingkatkan ke Log Amazon. CloudWatch

- MultiAZ— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan jika instans DB adalah deployment Multi-AZ.

- PendingModifiedValuesIni adalah sebuah PendingModifiedValues objek.

  Menentukan bahwa perubahan pada instans DB sedang tertunda. Elemen ini disertakan hanya ketika perubahan tertunda. Perubahan spesifik diidentifikasi oleh sub elemen.

- PreferredBackupWindow— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.

- PreferredMaintenanceWindow— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).

- PromotionTier— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Nilai yang menentukan urutan di mana Replika Baca dipromosikan ke instans primer setelah kegagalan instans primer yang ada.
- PubliclyAccessible— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Bendera ini seharusnya tidak lagi digunakan.
- ReadReplicaDBClusterIdentifiers— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi klaster DB yang merupakan Replika Baca instans DB ini.
- ReadReplicaDBInstanceIdentifiers— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu pengidentifikasi atau lebih dari Replika Baca yang terkait dengan instans DB ini.
- ReadReplicaSourceDBInstanceIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi instans DB sumber jika instans DB ini adalah Replika Baca.
- SecondaryAvailabilityZone— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jika ada, menentukan nama Availability Zone sekunder untuk instans DB dengan dukungan multi-AZ.
- StatusInfos— Ini adalah Array DBInstanceStatusInfo objek.

  Status Replika Baca. Jika instans bukan Replika Baca, ini kosong.
- StorageEncrypted— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Tidak didukung: Enkripsi untuk instans DB dikelola oleh klaster DB.
- StorageType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan jenis penyimpanan yang terkait dengan instans DB.
- TdeCredentialArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari penyimpanan kunci dengan yang terkait dengan instans untuk enkripsi TDE.

- Timezone— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung.

- VpcSecurityGroups— Ini adalah Array VpcSecurityGroupMembership objek.

  Menyediakan daftar elemen grup keamanan VPC tempat instans DB berada.

`DBInstance` digunakan sebagai elemen respon untuk:

- Buat DBInstance
- Hapus DBInstance
- Memodifikasi DBInstance
- Reboot DBInstance

DBInstanceStatusInfo (struktur)

Menyediakan daftar informasi status untuk sebuah instans DB.

Bidang

- Message— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Detail kesalahan jika ada kesalahan untuk instans. Jika instans tidak dalam status kesalahan, nilai ini kosong.

- Normal— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai Boolean yaitu true jika instans beroperasi secara normal, atau false jika instans berada dalam status kesalahan.

- Status— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Status dari instans DB. Untuk StatusType replika baca, nilainya dapat mereplikasi, kesalahan, berhenti, atau dihentikan.

- StatusType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nilai ini saat ini “replikasi baca.”

DBInstanceOpsi yang Dapat Dipesan (struktur)

Berisi daftar pilihan yang tersedia untuk instans DB.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called "DescribeOrderableDBInstancePilihan".

Bidang

- AvailabilityZones— Ini adalah Array AvailabilityZone objek.

  Daftar Availability Zone untuk instans DB.
- DBInstanceClass— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Kelas instans DB untuk instans DB.
- Engine— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis mesin dari sebuah instans DB.
- EngineVersion— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Versi mesin dari sebuah instans DB.
- LicenseModel— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Model lisensi untuk sebuah instans DB.
- MaxIopsPerDbInstance— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum Provisioned IOPS untuk instans DB.
- MaxIopsPerGib— Ini adalah DoubleOptional, dari tipe: `double` (nomor floating-point IEEE 754 presisi ganda).

  Maksimum Provisioned IOPS per GiB untuk instans DB.
- MaxStorageSize— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Ukuran penyimpanan maksimum untuk instans DB.
- MinIopsPerDbInstance— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

Jumlah minimum Provisioned IOPS untuk instans DB.

- MinIopsPerGib— Ini adalah DoubleOptional, dari tipe: `double` (nomor floating-point IEEE 754 presisi ganda).

  Minimal Provisioned IOPS per GiB untuk instans DB.

- MinStorageSize— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Ukuran penyimpanan minimum untuk instans DB.

- MultiAZCapable— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah instans DB mampu untuk Multi-AZ.

- ReadReplicaCapable— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah instans DB dapat memiliki Replica Baca.

- StorageType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak berlaku. Di Neptunus jenis penyimpanan dikelola pada tingkat DB Cluster.

- SupportsEnhancedMonitoring— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah instans DB mendukung Pemantauan Ditingkatkan pada interval mulai 1 sampai 60 detik.

- SupportsGlobalDatabases— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang menunjukkan apakah Anda dapat menggunakan database global Neptunus dengan kombinasi spesifik atribut mesin DB lainnya.

- SupportsIAMDatabaseAuthentication— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah instans DB mendukung autentikasi basis data IAM.

- SupportsIops— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah instans DB mendukung Provisioned IOPS.

- SupportsStorageEncryption— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

Menunjukkan apakah instans DB mendukung penyimpanan terenkripsi.

- Vpc— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah instans DB berada dalam VPC.

PendingModifiedValues (struktur)

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Memodifikasi DBInstance”.

Bidang

- AllocatedStorage— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Berisi ukuran `AllocatedStorage` baru untuk instans DB yang akan diterapkan atau sedang diterapkan.
- BackupRetentionPeriod— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang tertunda untuk penyimpanan cadangan otomatis dipertahankan.
- CACertificateIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan pengidentifikasi sertifikat (CA) untuk instans DB.
- DBInstanceClass— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi `DBInstanceClass` baru untuk instans DB yang akan diterapkan atau sedang diterapkan.
- DBInstanceIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi `DBInstanceIdentifier` baru untuk instans DB yang akan diterapkan atau sedang diterapkan.
- DBSubnetGroupName— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Grup subnet DB baru untuk instans DB.
- EngineVersion— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.

- Iops— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan nilai Provisioned IOPS baru untuk instans DB yang akan diterapkan atau sedang diterapkan.

- MultiAZ— Ini adalah BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan bahwa instans DB Single-AZ akan berubah menjadi deployment Multi-AZ.

- PendingCloudwatchLogsExportsIni adalah sebuah PendingCloudwatchLogsExports objek.

  `PendingCloudwatchLogsExports`Struktur ini menentukan perubahan yang tertunda untuk CloudWatch log mana yang diaktifkan dan mana yang dinonaktifkan.

- Port— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port tertunda untuk instans DB.

- StorageType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak berlaku. Di Neptunus jenis penyimpanan dikelola pada tingkat DB Cluster.

ValidStorageOptions (struktur)

Tidak berlaku. Di Neptunus jenis penyimpanan dikelola pada tingkat DB Cluster.

Bidang

- IopsToStorageRatio— Ini adalah Array DoubleRange objek.

  Tidak berlaku. Di Neptunus jenis penyimpanan dikelola pada tingkat DB Cluster.

- ProvisionedIops— Ini adalah Array Kisaran objek.

  Tidak berlaku. Di Neptunus jenis penyimpanan dikelola pada tingkat DB Cluster.

- StorageSize— Ini adalah Array Kisaran objek.

  Tidak berlaku. Di Neptunus jenis penyimpanan dikelola pada tingkat DB Cluster.

- StorageType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak berlaku. Di Neptunus jenis penyimpanan dikelola pada tingkat DB Cluster.

Valid DBInstance ModificationsMessage (struktur)

Informasi tentang modifikasi valid yang dapat Anda buat untuk instans DB Anda. Berisi hasil dari panggilan sukses ke tindakan the section called “DescribeValidDBInstanceModifikasi”. Anda dapat menggunakan informasi ini saat Anda memanggil the section called “Memodifikasi DBInstance”.

Bidang

- Storage— Ini adalah Array ValidStorageOptions objek.

  Opsi penyimpanan yang valid untuk instans DB Anda.

`ValidDBInstanceModificationsMessage` digunakan sebagai elemen respon untuk:

- DescribeValidDBInstanceModifikasi

API Parameter Neptune

Tindakan:

- Salin DBParameter Grup (tindakan)
- Salin DBCluster ParameterGroup (tindakan)
- Buat DBParameter Grup (aksi)
- Buat DBCluster ParameterGroup (tindakan)
- Hapus DBParameter Grup (tindakan)
- Hapus DBCluster ParameterGroup (tindakan)
- Ubah DBParameter Grup (tindakan)
- Memodifikasi DBCluster ParameterGroup (tindakan)
- Atur Ulang DBParameter Grup (tindakan)
- Atur ulang DBCluster ParameterGroup (tindakan)
- Jelaskan DBParameters (tindakan)
- Jelaskan DBParameter Grup (tindakan)
- Jelaskan DBCluster Parameter (tindakan)
- Jelaskan DBCluster ParameterGroups (tindakan)

- DescribeEngineDefaultParameters (tindakan)
- DescribeEngineDefaultClusterParameters (tindakan)

Struktur:

- Parameter (struktur)
- DBParameterKelompok (struktur)
- DBClusterParameterGroup (struktur)
- DBParameterGroupStatus (struktur)

Salin DBParameter Grup (tindakan)

Nama AWS CLI untuk API ini adalah:. `copy-db-parameter-group`

Menyalin grup parameter DB yang ditentukan.

Permintaan

- SourceDBParameterGroupIdentifier(dalam CLI:`--source-db-parameter-group-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengenal atau ARN untuk grup parameter DB sumber. Untuk informasi tentang pembuatan ARN, lihat Pembangunan Amazon Resource Name (ARN).

  Kendala:

  - Harus menentukan grup parameter DB yang valid.
  - Harus menentukan pengenal grup parameter DB yang valid, misalnya `my-db-param-group`, atau ARN yang valid.
- Tags(dalam CLI:`--tags`) — Sebuah array objek. Tag

  Tag yang ditetapkan ke grup parameter DB yang disalin.
- TargetDBParameterGroupDescription(dalam CLI:`--target-db-parameter-group-description`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Deskripsi untuk grup parameter DB yang disalin.
- TargetDBParameterGroupIdentifier(dalam CLI:`--target-db-parameter-group-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

Pengidentifikasi untuk grup parameter DByang disalin.

Kendala:

- Tidak dapat berupa null, kosong, atau blank.
- Harus berisi antara 1 hingga 255 huruf, angka, atau tanda hubung.
- Karakter pertama harus berupa huruf.
- Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.

Contoh: `my-db-parameter-group`

Respons

Berisi detail dari grup parameter Amazon Neptune DB.

Tipe data ini digunakan sebagai elemen respon dalam tindakan the section called “Jelaskan DBParameter Grup”.

- DBParameterGroupArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk grup parameter DB.
- DBParameterGroupFamily— String, tipe: `string` (string yang dikodekan UTF-8).

  Memberikan nama keluarga grup parameter DB yang kompatibel dengan grup parameter DB ini.
- DBParameterGroupName— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama grup parameter DB.
- Description— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan deskripsi yang ditentukan pelanggan untuk grup parameter DB ini.

Kesalahan

- DBParameterGroupNotFoundFault
- DBParameterGroupAlreadyExistsFault
- DBParameterGroupQuotaExceededFault

Salin DBCluster ParameterGroup (tindakan)

Nama AWS CLI untuk API ini adalah:. `copy-db-cluster-parameter-group`

Salinan grup parameter klaster DB tertentu.

Permintaan

- SourceDBClusterParameterGroupIdentifier(dalam CLI:`--source-db-cluster-parameter-group-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengenal atau Amazon Resource Name (ARN) untuk grup parameter klaster DB sumber. Untuk informasi tentang pembuatan ARN, lihat Pembangunan Amazon Resource Name (ARN).

  Kendala:

  - Harus menentukan grup parameter klaster DB yang valid.
  - Jika grup parameter klaster sumber berada dalam Wilayah Amazon yang sama dengan salinannya, tentukan pengenal grup parameter DB yang valid, misalnya `my-db-cluster-param-group`, atau ARN yang valid.
  - Jika grup parameter sumber berada dalam Wilayah Amazon yang berbeda dari salinannya, tentukan ARN grup parameter klaster DB yang valid, misalnya `arn:aws:rds:us-east-1:123456789012:cluster-pg:custom-cluster-group1`.
- Tags(dalam CLI:`--tags`) — Sebuah array objek. Tag

  Tag yang akan ditugaskan ke grup parameter klaster DB yang disalin.
- TargetDBClusterParameterGroupDescription(dalam CLI:`--target-db-cluster-parameter-group-description`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Deskripsi untuk grup parameter klaster DB yang disalin.
- TargetDBClusterParameterGroupIdentifier(dalam CLI:`--target-db-cluster-parameter-group-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengenal untuk grup parameter klaster DB yang disalin.

  Kendala:

  - Tidak dapat berupa null, kosong, atau blank
  - Harus berisi antara 1 hingga 255 huruf, angka, atau tanda hubung
  - Karakter pertamanya harus berupa huruf

- Tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung yang berurutan

  Contoh: `my-cluster-param-group1`

Respons

Berisi detail dari grup parameter klaster Amazon Neptune DB.

Tipe data ini digunakan sebagai elemen respon dalam tindakan the section called “Jelaskan DBCluster ParameterGroups”.

- DBClusterParameterGroupArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk grup parameter klaster DB.
- DBClusterParameterGroupName— String, tipe: `string` (string yang dikodekan UTF-8).

  Memberikan nama grup parameter klaster DB.
- DBParameterGroupFamily— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama keluarga grup parameter DB yang kompatibel dengan grup parameter klaster DB ini.
- Description— String, tipe: `string` (string yang dikodekan UTF-8).

  Memberikan deskripsi khusus pelanggan untuk grup parameter klaster DB ini.

Kesalahan

- DBParameterGroupNotFoundFault
- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

Buat DBParameter Grup (aksi)

Nama AWS CLI untuk API ini adalah:. `create-db-parameter-group`

Membuat grup parameter DB baru.

Grup parameter DB awalnya dibuat dengan parameter default untuk mesin basis data yang digunakan oleh instans DB. Untuk memberikan nilai kustom untuk salah satu parameter, Anda harus

memodifikasi grup setelah membuatnya menggunakan Modify DBParameter Group. Setelah Anda membuat grup parameter DB, Anda perlu mengaitkannya dengan instans DB Anda menggunakan Modify DBInstance. Ketika Anda mengaitkan grup parameter DB baru dengan instans DB yang berjalan, Anda perlu but ulang instans DB tanpa failover untuk grup parameter DB baru dan pengaturan terkait agar berlaku.

> **Important**
>
> Setelah Anda membuat grup parameter DB, Anda harus menunggu setidaknya 5 menit sebelum membuat instans DB pertama Anda yang menggunakan grup parameter DB tersebut sebagai grup parameter default. Ini memungkinkan Amazon Neptune untuk sepenuhnya menyelesaikan tindakan pembuatan sebelum grup parameter digunakan sebagai default untuk instans DB baru. Ini penting khususnya untuk parameter yang sangat penting saat membuat basis data default untuk sebuah instans DB, seperti set karakter untuk basis data default yang ditentukan oleh parameter `character_set_database`. Anda dapat menggunakan opsi Grup Parameter konsol Amazon Neptunus atau perintah DBParameters Deskripsikan untuk memverifikasi bahwa grup parameter DB Anda telah dibuat atau dimodifikasi.

Permintaan

- DBParameterGroupFamily(dalam CLI:`--db-parameter-group-family`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama keluarga grup parameter DB. Grup parameter DB dapat dikaitkan dengan satu dan hanya satu keluarga grup parameter DB, dan hanya dapat diterapkan ke instans DB yang menjalankan mesin basis data dan versi mesin yang kompatibel dengan keluarga grup parameter DB tersebut.
- DBParameterGroupName(dalam CLI:`--db-parameter-group-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter DB.

  Kendala:

  - Harus berisi 1 sampai 255 huruf, angka, atau tanda hubung.
  - Karakter pertamanya harus berupa huruf
  - Tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung yang berurutan

> **Note**
>
> Nilai ini disimpan sebagai string huruf kecil.

- Description(dalam CLI:`--description`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Deskripsi untuk grup parameter DB.

- Tags(dalam CLI:`--tags`) — Sebuah array objek. Tag

  Tag yang akan ditugaskan ke grup parameter DB baru.

Respons

Berisi detail dari grup parameter Amazon Neptune DB.

Tipe data ini digunakan sebagai elemen respon dalam tindakan the section called “Jelaskan DBParameter Grup”.

- DBParameterGroupArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk grup parameter DB.

- DBParameterGroupFamily— String, tipe: `string` (string yang dikodekan UTF-8).

  Memberikan nama keluarga grup parameter DB yang kompatibel dengan grup parameter DB ini.

- DBParameterGroupName— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama grup parameter DB.

- Description— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan deskripsi yang ditentukan pelanggan untuk grup parameter DB ini.

Kesalahan

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

Buat DBCluster ParameterGroup (tindakan)

Nama AWS CLI untuk API ini adalah:. `create-db-cluster-parameter-group`

Membuat grup parameter klaster DB baru.

Parameter dalam grup parameter klaster DB berlaku untuk semua instans dalam sebuah klaster DB.

Grup parameter klaster DB awalnya dibuat dengan parameter default untuk mesin basis data yang digunakan oleh instans dalam klaster DB. Untuk menyediakan nilai kustom bagi parameter apa pun, Anda harus mengubah grup setelah membuatnya menggunakan the section called “Memodifikasi DBCluster ParameterGroup”. Setelah Anda membuat grup parameter klaster DB, Anda perlu mengaitkannya dengan klaster DB Anda menggunakan the section called “Memodifikasi DBCluster”. Ketika Anda mengaitkan grup parameter klaster DB baru dengan klaster DB yang berjalan, Anda perlu melakukan reboot instans DB di dalam klaster DB tanpa failover untuk grup parameter klaster DB baru dan pengaturan terkait agar berlaku.

> **Important**
>
> Setelah Anda membuat grup parameter klaster DB, Anda harus menunggu setidaknya 5 menit sebelum membuat klaster DB pertama Anda yang menggunakan grup parameter klaster DB tersebut sebagai grup parameter default. Ini memungkinkan Amazon Neptune untuk sepenuhnya menyelesaikan tindakan pembuatan sebelum grup parameter klaster DB digunakan sebagai default untuk klaster DB baru. Ini penting khususnya untuk parameter yang sangat penting saat membuat basis data default untuk sebuah klaster DB, seperti set karakter untuk basis data default yang ditentukan oleh parameter `character_set_database`. Anda dapat menggunakan pilihan Grup Parameter dari Amazon Neptune console atau perintahthe section called “Jelaskan DBCluster Parameter” untuk memverifikasi bahwa grup parameter klaster DB Anda telah dibuat atau diubah.

Permintaan

- DBClusterParameterGroupName(dalam CLI:`--db-cluster-parameter-group-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter klaster.

  Batasan:

- Harus cocok dengan nama yang sudah ada DBClusterParameterGroup.

> **Note**
>
> Nilai ini disimpan sebagai string huruf kecil.

- DBParameterGroupFamily(dalam CLI:`--db-parameter-group-family`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama keluarga grup parameter klaster DB. Grup parameter DB dapat dikaitkan dengan satu dan hanya satu keluarga grup parameter klaster DB, dan hanya dapat diterapkan ke klaster DB yang menjalankan mesin basis data dan versi mesin yang kompatibel dengan keluarga grup parameter klaster DB tersebut.
- Description(dalam CLI:`--description`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Deskripsi untuk grup parameter klaster DB.
- Tags(dalam CLI:`--tags`) — Sebuah array objek. Tag

  Tag yang akan ditugaskan ke grup parameter kluster DB baru.

Respons

Berisi detail dari grup parameter klaster Amazon Neptune DB.

Tipe data ini digunakan sebagai elemen respon dalam tindakan the section called “Jelaskan DBCluster ParameterGroups”.

- DBClusterParameterGroupArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk grup parameter klaster DB.
- DBClusterParameterGroupName— String, tipe: `string` (string yang dikodekan UTF-8).

  Memberikan nama grup parameter klaster DB.
- DBParameterGroupFamily— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama keluarga grup parameter DB yang kompatibel dengan grup parameter klaster DB ini.
- Description— String, tipe: `string` (string yang dikodekan UTF-8).

Memberikan deskripsi khusus pelanggan untuk grup parameter klaster DB ini.

Kesalahan

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

Hapus DBParameter Grup (tindakan)

Nama AWS CLI untuk API ini adalah:. `delete-db-parameter-group`

Menghapus DBParameter Grup tertentu. DBParameterGrup yang akan dihapus tidak dapat dikaitkan dengan instans DB apa pun.

Permintaan

- DBParameterGroupName(dalam CLI:`--db-parameter-group-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter DB.

  Kendala:

  - Harus nama grup parameter DB yang ada
  - Anda tidak dapat menhapus grup parameter DB.
  - Tidak dapat dikaitkan dengan instans DB apapun

Respons

- Tidak ada parameter Respons.

Kesalahan

- Tidak valid DBParameter GroupStateFault
- DBParameterGroupNotFoundFault

Hapus DBCluster ParameterGroup (tindakan)

Nama AWS CLI untuk API ini adalah:. `delete-db-cluster-parameter-group`

Menghapus grup parameter klaster DB tertentu. Grup parameter klaster DB yang akan dihapus tidak dapat dikaitkan dengan klaster DB apa pun.

Permintaan

- DBClusterParameterGroupName(dalam CLI:`--db-cluster-parameter-group-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter klaster.

  Kendala:

  - Harus nama grup parameter kluster DB yang sudah ada.
  - Anda tidak dapat menghapus grup parameter klaster DB default.
  - Tidak dapat dikaitkan dengan klaster DB mana pun.

Respons

- Tidak ada parameter Respons.

Kesalahan

- Tidak valid DBParameter GroupStateFault
- DBParameterGroupNotFoundFault

Ubah DBParameter Grup (tindakan)

Nama AWS CLI untuk API ini adalah:. `modify-db-parameter-group`

Memodifikasi parameter dari grup parameter DB. Untuk mengubah lebih dari satu parameter, kirimkan daftar berikut ini: `ParameterName`, `ParameterValue`, and `ApplyMethod`. Maksimum 20 parameter dapat dimodifikasi dalam satu permintaan.

> **Note**
>
> Perubahan pada parameter dinamis langsung diterapkan. Perubahan parameter statis memerlukan reboot tanpa failover ke instans DB yang terkait dengan grup parameter sebelum perubahan dapat berlaku.

> **Important**
>
> Setelah Anda mengubah grup parameter DB, tunggu setidaknya 5 menit sebelum membuat instans DB pertama Anda yang menggunakan grup parameter DB tersebut sebagai grup parameter default. Ini memungkinkan Amazon Neptune untuk sepenuhnya menyelesaikan tindakan pembuatan sebelum grup parameter digunakan sebagai default untuk instans DB baru. Ini penting khususnya untuk parameter yang sangat penting saat membuat basis data default untuk sebuah instans DB, seperti set karakter untuk basis data default yang ditentukan oleh parameter `character_set_database`. Anda dapat menggunakan opsi Grup Parameter konsol Amazon Neptunus atau perintah DBParameters Deskripsikan untuk memverifikasi bahwa grup parameter DB Anda telah dibuat atau dimodifikasi.

Permintaan

- DBParameterGroupName(dalam CLI:`--db-parameter-group-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter DB.

  Batasan:

  - Jika disediakan, harus cocok dengan nama DBParameter Grup yang ada.
- Parameters(dalam CLI:`--parameters`) - Diperlukan: Sebuah array objek. Parameter

  Array nama parameter, nilai dan metode penerapan untuk pembaruan parameter. Setidaknya satu nama parameter, nilai, dan metode penerapan harus disediakan; argumen berikutnya adalah opsional. Maksimum 20 parameter dapat dimodifikasi dalam satu permintaan.

  Nilai yang Valid (untuk metode aplikasi): `immediate | pending-reboot`

> **Note**
>
> Anda dapat menggunakan nilai langsung dengan parameter dinamis saja. Anda dapat menggunakan nilai pending-reboot untuk parameter dinamis dan statis, dan perubahan diterapkan ketika Anda melakukan reboot instans DB tanpa failover.

Respons

- DBParameterGroupName— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama grup parameter DB.

Kesalahan

- DBParameterGroupNotFoundFault
- Tidak valid DBParameter GroupStateFault

Memodifikasi DBCluster ParameterGroup (tindakan)

Nama AWS CLI untuk API ini adalah:. `modify-db-cluster-parameter-group`

Memodifikasi parameter dari grup parameter klaster DB. Untuk mengubah lebih dari satu parameter, kirimkan daftar berikut ini: `ParameterName`, `ParameterValue`, and `ApplyMethod`. Maksimum 20 parameter dapat dimodifikasi dalam satu permintaan.

> **Note**
>
> Perubahan pada parameter dinamis langsung diterapkan. Perubahan ke parameter statis memerlukan reboot tanpa failover ke cluster DB terkait dengan grup parameter sebelum perubahan dapat berlaku.

> **Important**
>
> Setelah Anda membuat grup parameter klaster DB, Anda harus menunggu setidaknya 5 menit sebelum membuat klaster DB pertama Anda yang menggunakan grup parameter

klaster DB tersebut sebagai grup parameter default. Ini mengizinkan Amazon Neptune dapat menyelesaikan sepenuhnya tindakan pembuatan sebelum grup parameter digunakan sebagai default untuk klaster DB baru. Ini penting khususnya untuk parameter yang sangat penting saat membuat basis data default untuk sebuah klaster DB, seperti set karakter untuk basis data default yang ditentukan oleh parameter `character_set_database`. Anda dapat menggunakan pilihan Grup Parameter dari konsol Amazon Neptune atau perintah the section called “Jelaskan DBCluster Parameter” untuk memverifikasi bahwa grup parameter klaster DB Anda telah dibuat atau diubah.

Permintaan

- DBClusterParameterGroupName(dalam CLI:`--db-cluster-parameter-group-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter klaster DB untuk dimodifikasi.
- Parameters(dalam CLI:`--parameters`) - Diperlukan: Sebuah array objek. Parameter

  Daftar parameter dalam grup parameter klaster DB untuk dimodifikasi.

Respons

- DBClusterParameterGroupName— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter klaster.

  Kendala:
  - Harus huruf atau angka berisi 1 sampai 255 karakter.
  - Karakter pertamanya harus berupa huruf
  - Tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung yang berurutan

> **Note**
>
> Nilai ini disimpan sebagai string huruf kecil.

Kesalahan

- DBParameterGroupNotFoundFault
- Tidak valid DBParameter GroupStateFault

Atur Ulang DBParameter Grup (tindakan)

Nama AWS CLI untuk API ini adalah:. `reset-db-parameter-group`

Memodifikasi parameter grup parameter DB ke nilai engine/system default. Untuk mereset parameter tertentu, kirimkan daftar berikut ini: `ParameterName` dan `ApplyMethod`. Untuk mereset seluruh grup parameter DB, tentukan nama `DBParameterGroup` dan parameter `ResetAllParameters`. Ketika Anda mereset seluruh grup, parameter dinamis segera diperbarui dan parameter statis diatur ke `pending-reboot` untuk berlaku pada restart instans DB berikutnya atau permintaan `RebootDBInstance`

Permintaan

- DBParameterGroupName(dalam CLI:`--db-parameter-group-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter DB.

  Batasan:

  - Harus cocok dengan nama DBParameter Grup yang ada.
- Parameters(dalam CLI:`--parameters`) — Sebuah array objek. Parameter

  Untuk mereset seluruh grup parameter DB, tentukan nama `DBParameterGroup` dan parameter `ResetAllParameters`. Untuk mereset parameter tertentu, kirimkan daftar berikut ini: `ParameterName` dan `ApplyMethod`. Maksimum 20 parameter dapat dimodifikasi dalam satu permintaan.

  Nilai yang Valid (untuk metode Penerapan): `pending-reboot`
- ResetAllParameters(dalam CLI:`--reset-all-parameters`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah (`true`) atau tidak (`false`) akan mereset semua parameter di grup parameter DB ke nilai defaultnya.

Default: `true`

Respons

- DBParameterGroupName— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama grup parameter DB.

Kesalahan

- Tidak valid DBParameter GroupStateFault
- DBParameterGroupNotFoundFault

Atur ulang DBCluster ParameterGroup (tindakan)

Nama AWS CLI untuk API ini adalah:. `reset-db-cluster-parameter-group`

Memodifikasi parameter dari grup parameter klaster DB ke nilai default. Untuk mereset parameter tertentu, kirimkan daftar berikut ini: `ParameterName` dan `ApplyMethod`. Untuk mereset seluruh grup parameter klaster DB, tentukan parameter `DBClusterParameterGroupName` dan `ResetAllParameters`.

Ketika Anda mereset seluruh grup, parameter dinamis segera diperbarui dan parameter statis diatur ke `pending-reboot` untuk berlaku pada restart instans DB berikutnya atau permintaan the section called “Reboot DBInstance” Anda harus menelepon the section called “Reboot DBInstance” untuk setiap instans DB di klaster DB Anda yang Anda ingin parameter statis diperbaruinya diterapkan.

Permintaan

- DBClusterParameterGroupName(dalam CLI:`--db-cluster-parameter-group-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter klaster DB untuk direset.
- Parameters(dalam CLI:`--parameters`) — Sebuah array objek. Parameter

  Daftar nama parameter dalam grup parameter klaster DB untuk direset ke nilai default. Anda tidak dapat menggunakan parameter ini jika parameter `ResetAllParameters` diatur ke `true`.

- ResetAllParameters(dalam CLI:`--reset-all-parameters`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang diatur ke `true` untuk me-reset semua parameter dalam kelompok parameter klaster DB ke nilai defaultnya, dan `false` sebaliknya. Anda tidak dapat menggunakan parameter ini jika ada daftar nama parameter yang ditentukan untuk parameter `Parameters`.

Respons

- DBClusterParameterGroupName— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter klaster.

  Kendala:

  - Harus huruf atau angka berisi 1 sampai 255 karakter.
  - Karakter pertamanya harus berupa huruf
  - Tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung yang berurutan

  > **Note**
  >
  > Nilai ini disimpan sebagai string huruf kecil.

Kesalahan

- Tidak valid DBParameter GroupStateFault
- DBParameterGroupNotFoundFault

Jelaskan DBParameters (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-db-parameters`

Mengembalikan daftar parameter detail untuk grup parameter DB khusus.

Permintaan

- DBParameterGroupName(dalam CLI:`--db-parameter-group-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

Nama grup parameter DB tertentu untuk mengembalikan detail.

Batasan:

- Jika disediakan, harus cocok dengan nama DBParameter Grup yang ada.

- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Parameter ini saat ini tidak didukung.

- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan `DescribeDBParameters` sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

  Default: 100

  Kendala: Minimal 20, maksimum 100.

- Source(dalam CLI:`--source`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis parameter untuk dikembalikan.

  Default: Semua jenis parameter yang dikembalikan

  Nilai Valid: `user | system | engine-default`

Respons

- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

- Parameters – Susunan objek Parameter.

Daftar nilai the section called “Parameter”.

Kesalahan

- DBParameterGroupNotFoundFault

Jelaskan DBParameter Grup (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-db-parameter-groups`

Mengembalikan daftar deskripsi `DBParameterGroup`. Jika `DBParameterGroupName` ditentukan, daftar hanya akan berisi deskripsi dari grup parameter DB yang ditentukan.

Permintaan

- DBParameterGroupName(dalam CLI:`--db-parameter-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter DB tertentu untuk mengembalikan detail.

  Batasan:

  - Jika disediakan, harus cocok dengan nama yang sudah ada DBClusterParameterGroup.
- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Parameter ini saat ini tidak didukung.
- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan `DescribeDBParameterGroups` sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.
- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

  Default: 100

Kendala: Minimal 20, maksimum 100.

Respons

- DBParameterGroups – Susunan objek DBParameterKelompok.

  Daftar instans the section called “DBParameterKelompok”.

- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

Kesalahan

- DBParameterGroupNotFoundFault

Jelaskan DBCluster Parameter (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-db-cluster-parameters`

Mengembalikan daftar parameter detail untuk grup parameter kluster DB tertentu.

Permintaan

- DBClusterParameterGroupName(dalam CLI:`--db-cluster-parameter-group-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter klaster DB tertentu yang detail parameternya dikembalikan.

  Batasan:

  - Jika disediakan, harus cocok dengan nama yang sudah ada DBClusterParameterGroup.

- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Parameter ini saat ini tidak didukung.

- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

Token pagination opsional yang disediakan oleh permintaan `DescribeDBClusterParameters` sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

  Default: 100

  Kendala: Minimal 20, maksimum 100.

- Source(dalam CLI:`--source`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nilai yang menunjukkan untuk mengembalikan hanya parameter untuk sumber tertentu. Sumber parameter dapat berupa `engine`, `service`, atau `customer`.

Respons

- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan Deskripsikan DBCluster Parameter sebelumnya. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

- Parameters – Susunan objek Parameter.

  Menyediakan daftar parameter untuk grup parameter klaster DB.

Kesalahan

- DBParameterGroupNotFoundFault

Jelaskan DBCluster ParameterGroups (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-db-cluster-parameter-groups`

Mengembalikan daftar deskripsi `DBClusterParameterGroup`. Jika parameter `DBClusterParameterGroupName` ditentukan, daftar berisi hanya deskripsi grup parameter klaster DB yang ditentukan.

Permintaan

- DBClusterParameterGroupName(dalam CLI:`--db-cluster-parameter-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter klaster DB tertentu yang detailnya dikembalikan.

  Batasan:

  - Jika disediakan, harus cocok dengan nama yang sudah ada DBClusterParameterGroup.
- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Parameter ini saat ini tidak didukung.
- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan `DescribeDBClusterParameterGroups` sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.
- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

  Default: 100

  Kendala: Minimal 20, maksimum 100.

Respons

- DBClusterParameterGroups – Susunan objek DBClusterParameterGroup.

  Daftar grup parameter klaster DB.
- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

Token pagination opsional yang disediakan oleh permintaan `DescribeDBClusterParameterGroups` sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

Kesalahan

- DBParameterGroupNotFoundFault

DescribeEngineDefaultParameters (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-engine-default-parameters`

Mengembalikan mesin default dan informasi parameter sistem untuk mesin basis data tertentu.

Permintaan

- DBParameterGroupFamily(dalam CLI:`--db-parameter-group-family`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama keluarga grup parameter DB.
- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Saat ini tidak didukung.
- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan `DescribeEngineDefaultParameters` sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.
- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

  Default: 100

  Kendala: Minimal 20, maksimum 100.

Respons

Berisi hasil pemanggilan yang berhasil dari tindakan the section called "DescribeEngineDefaultParameters".

- DBParameterGroupFamily— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama keluarga grup parameter DB yang diterapkan parameter default mesin.
- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh EngineDefaults permintaan sebelumnya. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.
- Parameters – Susunan objek Parameter.

  Berisi daftar parameter default mesin.

DescribeEngineDefaultClusterParameters (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-engine-default-cluster-parameters`

Mengembalikan mesin default dan informasi parameter sistem untuk mesin basis data klaster.

Permintaan

- DBParameterGroupFamily(dalam CLI:`--db-parameter-group-family`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama keluarga grup parameter klaster DB untuk mengembalikan informasi parameter mesin.
- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Parameter ini saat ini tidak didukung.
- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan `DescribeEngineDefaultClusterParameters` sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

  Default: 100

  Kendala: Minimal 20, maksimum 100.

Respons

Berisi hasil pemanggilan yang berhasil dari tindakan the section called "DescribeEngineDefaultParameters".

- DBParameterGroupFamily— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama keluarga grup parameter DB yang diterapkan parameter default mesin.
- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh EngineDefaults permintaan sebelumnya. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.
- Parameters – Susunan objek Parameter.

  Berisi daftar parameter default mesin.

Struktur:

Parameter (struktur)

Menentukan parameter.

Bidang

- AllowedValues— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang nilai yang valid untuk parameter.

- ApplyMethod— Ini adalah ApplyMethod, dari tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan kapan harus menerapkan pembaruan parameter.
- ApplyType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan mesin tipe parameter tertentu.
- DataType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan tipe data yang valid untuk parameter.
- Description— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan deskripsi parameter.
- IsModifiable— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah (`true`) atau tidak (`false`) parameter dapat dimodifikasi. Beberapa parameter memiliki implikasi keamanan atau operasional yang mencegahnya diubah.
- MinimumEngineVersion— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Versi mesin paling awal yang parameternya dapat diterapkan.
- ParameterName— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama dari parameter.
- ParameterValue— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nilai dari parameter.
- Source— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan sumber nilai parameter.

DBParameterKelompok (struktur)

Berisi detail dari grup parameter Amazon Neptune DB.

Tipe data ini digunakan sebagai elemen respon dalam tindakan the section called “Jelaskan DBParameter Grup”.

Bidang

- DBParameterGroupArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

Amazon Resource Name (ARN) untuk grup parameter DB.

- DBParameterGroupFamily— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Memberikan nama keluarga grup parameter DB yang kompatibel dengan grup parameter DB ini.
- DBParameterGroupName— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama grup parameter DB.
- Description— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Memberikan deskripsi yang ditentukan pelanggan untuk kelompok parameter DB ini.

`DBParameterGroup` digunakan sebagai elemen respon untuk:

- Salin DBParameter Grup
- Buat DBParameter Grup

DBClusterParameterGroup (struktur)

Berisi detail dari grup parameter klaster Amazon Neptune DB.

Tipe data ini digunakan sebagai elemen respon dalam tindakan the section called “Jelaskan DBCluster ParameterGroups”.

Bidang

- DBClusterParameterGroupArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk grup parameter klaster DB.
- DBClusterParameterGroupName— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Memberikan nama grup parameter klaster DB.
- DBParameterGroupFamily— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama keluarga grup parameter DB yang kompatibel dengan grup parameter klaster DB ini.
- Description— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan deskripsi yang ditentukan pelanggan untuk grup parameter klaster DB ini.

`DBClusterParameterGroup` digunakan sebagai elemen respon untuk:

- Salin DBCluster ParameterGroup
- Buat DBCluster ParameterGroup

DBParameterGroupStatus (struktur)

Status grup parameter DB..

Tipe data ini digunakan sebagai elemen respon dalam tindakan berikut:

- the section called “Buat DBInstance”
- the section called “Hapus DBInstance”
- the section called “Memodifikasi DBInstance”
- the section called “Reboot DBInstance”

Bidang

- DBParameterGroupName— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter DP.
- ParameterApplyStatus— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Status pembaruan parameter.

API Subnet Neptune

Tindakan:

- Buat DBSubnet Grup (aksi)
- Hapus DBSubnet Grup (tindakan)
- Ubah DBSubnet Grup (tindakan)
- Jelaskan DBSubnet Grup (tindakan)

Struktur:

- Subnet (struktur)
- DBSubnetKelompok (struktur)

Buat DBSubnet Grup (aksi)

Nama AWS CLI untuk API ini adalah:. `create-db-subnet-group`

Membuat grup subnet DB baru. Grup subnet DB harus berisi setidaknya satu subnet di setidaknya dua subnet AZs di Wilayah Amazon.

Permintaan

- DBSubnetGroupDescription(dalam CLI:`--db-subnet-group-description`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Deskripsi untuk grup subnet DB.
- DBSubnetGroupName(dalam CLI:`--db-subnet-group-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Naman grup subnet DB. Nilai ini disimpan sebagai string huruf kecil.

  Kendala: Harus berisi tidak lebih dari 255 huruf, angka, titik, garis bawah, spasi, atau tanda hubung. Tidak harus default.

  Contoh: `mySubnetgroup`
- SubnetIds(dalam CLI:`--subnet-ids`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  EC2 Subnet IDs untuk grup subnet DB.
- Tags(dalam CLI:`--tags`) — Sebuah array objek. Tag

  Tanda yang akan ditetapkan ke grup subnet DB.

Respons

Berisi detail dari grup subnet Amazon Neptune DB.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBSubnet Grup”

- DBSubnetGroupArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk grup subnet DB.
- DBSubnetGroupDescription— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan deskripsi grup subnet DB.
- DBSubnetGroupName— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup subnet DB.
- SubnetGroupStatus— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan status grup subnet DB.
- Subnets – Susunan objek Subnet.

  Berisi daftar elemen the section called “Subnet”.
- VpcId— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan VpcId grup subnet DB.

Kesalahan

- DBSubnetGroupAlreadyExistsFault
- DBSubnetGroupQuotaExceededFault
- DBSubnetQuotaExceededFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

Hapus DBSubnet Grup (tindakan)

Nama AWS CLI untuk API ini adalah:. `delete-db-subnet-group`

Menghapus grup subnet DB.

> **Note**
>
> Grup subnet basis data yang ditentukan tidak boleh dikaitkan dengan instans DB apa pun.

Permintaan

- DBSubnetGroupName(dalam CLI:`--db-subnet-group-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup subnet basis data yang akan dihapus.

  > **Note**
  >
  > Anda tidak dapat menghapus grup subnet default.

  Batasan:

  Kendala: Harus cocok dengan nama Grup yang ada. DBSubnet Tidak harus default.

  Contoh: `mySubnetgroup`

Respons

- Tidak ada parameter Respons.

Kesalahan

- Tidak valid DBSubnet GroupStateFault
- Tidak valid DBSubnet StateFault
- DBSubnetGroupNotFoundFault

Ubah DBSubnet Grup (tindakan)

Nama AWS CLI untuk API ini adalah:. `modify-db-subnet-group`

Mengubah grup subnet DB yang ada. Grup subnet DB harus berisi setidaknya satu subnet di setidaknya dua subnet AZs di Wilayah Amazon.

Permintaan

- DBSubnetGroupDescription(dalam CLI:`--db-subnet-group-description`) — String, tipe: `string` (string yang dikodekan UTF-8).

Deskripsi untuk grup subnet DB.

- DBSubnetGroupName(dalam CLI:`--db-subnet-group-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Naman grup subnet DB. Nilai ini disimpan sebagai string huruf kecil. Anda tidak dapat memodifikasi grup subnet default.

  Kendala: Harus cocok dengan nama Grup yang ada. DBSubnet Tidak harus default.

  Contoh: `mySubnetgroup`
- SubnetIds(dalam CLI:`--subnet-ids`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  EC2 Subnet IDs untuk grup subnet DB.

Respons

Berisi detail dari grup subnet Amazon Neptune DB.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBSubnet Grup”

- DBSubnetGroupArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk grup subnet DB.
- DBSubnetGroupDescription— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan deskripsi grup subnet DB.
- DBSubnetGroupName— String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup subnet DB.
- SubnetGroupStatus— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan status grup subnet DB.
- Subnets – Susunan objek Subnet.

  Berisi daftar elemen the section called “Subnet”.
- VpcId— String, tipe: `string` (string yang dikodekan UTF-8).

Menyediakan VpcId grup subnet DB.

Kesalahan

- DBSubnetGroupNotFoundFault
- DBSubnetQuotaExceededFault
- SubnetAlreadyInUse
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

Jelaskan DBSubnet Grup (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-db-subnet-groups`

Mengembalikan daftar deskripsi DBSubnet Grup. Jika DBSubnet GroupName ditentukan, daftar hanya akan berisi deskripsi dari DBSubnet Grup yang ditentukan.

Untuk gambaran umum rentang CIDR, buka Tutorial Wikipedia.

Permintaan

- DBSubnetGroupName(dalam CLI:`--db-subnet-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup subnet DB yang akan dikembalikan detailnya.
- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Parameter ini saat ini tidak didukung.
- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan Describe DBSubnet Groups sebelumnya. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.
- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

Default: 100

Kendala: Minimal 20, maksimum 100.

Respons

- DBSubnetGroups – Susunan objek DBSubnetKelompok.

  Daftar instans the section called “DBSubnetKelompok”.
- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

Kesalahan

- DBSubnetGroupNotFoundFault

Struktur:

Subnet (struktur)

Menentukan subnet.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBSubnet Grup”

Bidang

- SubnetAvailabilityZoneIni adalah sebuah AvailabilityZone objek.

  Menentukan EC2 Availability Zone yang subnet berada di.
- SubnetIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

Menentukan pengidentifikasi dari subnet.

- SubnetStatus— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status dari subnet.

DBSubnetKelompok (struktur)

Berisi detail dari grup subnet Amazon Neptune DB.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBSubnet Grup”

Bidang

- DBSubnetGroupArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk grup subnet DB.
- DBSubnetGroupDescription— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan deskripsi grup subnet DB.
- DBSubnetGroupName— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup subnet DB.
- SubnetGroupStatus— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan status grup subnet DB.
- Subnets— Ini adalah Array Subnet objek.

  Berisi daftar elemen the section called “Subnet”.
- VpcId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan VpcId grup subnet DB.

`DBSubnetGroup` digunakan sebagai elemen respons untuk:

- Buat DBSubnet Grup
- Ubah DBSubnet Grup

API Snapshot Neptune

Tindakan:

- Buat DBCluster Snapshot (aksi)
- Hapus DBCluster Snapshot (tindakan)
- Salin DBCluster Snapshot (tindakan)
- Memodifikasi DBCluster SnapshotAttribute (tindakan)
- Kembalikan DBCluster FromSnapshot (tindakan)
- Kembalikan DBCluster ToPointInTime (tindakan)
- Jelaskan DBCluster Snapshots (tindakan)
- Jelaskan DBCluster SnapshotAttributes (tindakan)

Struktur:

- DBClusterSnapshot (struktur)
- DBClusterSnapshotAttribute (struktur)
- DBClusterSnapshotAttributesResult (struktur)

Buat DBCluster Snapshot (aksi)

> Nama AWS CLI untuk API ini adalah:. `create-db-cluster-snapshot`

Membuat snapshot klaster DB

Permintaan

- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi klaster DB untuk membuat snapshotnya. Parameter ini tidak peka huruf besar kecil.

  Batasan:

  - Harus cocok dengan pengenal yang sudah ada DBCluster.

Contoh: `my-cluster1`

- DBClusterSnapshotIdentifier(dalam CLI:`--db-cluster-snapshot-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi snapshot klaster DB. Parameter ini disimpan sebagai string huruf kecil.

  Batas:

  - Harus berisi 1 sampai 63 huruf, angka, atau tanda hubung.
  - Karakter pertama harus berupa huruf.
  - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.

  Contoh: `my-cluster1-snapshot1`
- Tags(dalam CLI:`--tags`) — Sebuah array objek. Tag

  Tanda yang akan ditetapkan ke snapshot klaster DB.

Respons

Berisi detail untuk snapshot klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBCluster Snapshots”.

- AllocatedStorage— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan ukuran penyimpanan yang dialokasikan dalam gibibytes (GiB).
- AvailabilityZones— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance dalam snapshot cluster DB dapat dipulihkan.
- ClusterCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).
- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan pengidentifikasi klaster DB dari klaster DB tempat snapshot klaster DB ini dibuat.
- DBClusterSnapshotArn— String, tipe: `string` (string yang dikodekan UTF-8).

Amazon Resource Name (ARN) untuk snapshot klaster DB.

- DBClusterSnapshotIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan pengidentifikasi untuk snapshot klaster DB. Harus cocok dengan pengidentifikasi snapshot yang ada.

  Setelah Anda memulihkan klaster DB menggunakan `DBClusterSnapshotIdentifier`, Anda harus menentukan `DBClusterSnapshotIdentifier` yang sama untuk pembaruan klaster DB di masa mendatang. Ketika Anda menentukan properti ini untuk pembaruan, klaster DB tidak dipulihkan dari snapshot lagi, dan data dalam basis data tidak berubah.

  Namun, jika Anda tidak menentukan `DBClusterSnapshotIdentifier`, klaster DB kosong akan dibuat, dan klaster DB awal dihapus. Jika Anda menentukan properti yang berbeda dari properti pemulihan snapshot sebelumnya, klaster DB dipulihkan dari snapshot yang ditentukan oleh `DBClusterSnapshotIdentifier`, dan klaster DB awal dihapus.
- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama mesin basis data.
- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan versi mesin basis data untuk snapshot klaster DB ini.
- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.
- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah true, pengidentifikasi kunci Amazon KMS untuk snapshot klaster DB terenkripsi.
- LicenseModel— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan informasi model lisensi untuk snapshot klaster DB ini.
- PercentProgress— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan persentase perkiraan data yang telah ditransfer.
- Port— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

Menentukan port yang didengarkan klaster DB pada waktu dari snapshot.

- SnapshotCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menyediakan waktu ketika snapshot diambil, dalam Universal Coordinated Time (UTC).
- SnapshotType— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan jenis snapshot klaster DB.
- SourceDBClusterSnapshotArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika snapshot klaster DB disalin dari snapshot klaster DB sumber, Amazon Resource Name (ARN) untuk snapshot klaster DB sumber; jika tidak, nilai null.
- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status snapshot klaster DB ini.
- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah snapshot klaster DB dienkripsi.
- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang terkait dengan snapshot cluster DB.
- VpcId— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan ID VPC terkait dengan snapshot klaster DB.

Kesalahan

- DBClusterSnapshotAlreadyExistsFault
- Tidak valid DBCluster StateFault
- DBClusterNotFoundFault
- SnapshotQuotaExceededFault
- Tidak valid DBCluster SnapshotStateFault

Hapus DBCluster Snapshot (tindakan)

Nama AWS CLI untuk API ini adalah:. `delete-db-cluster-snapshot`

Menghapus snapshot klaster DB. Jika snapshot sedang disalin, operasi penyalinan dihentikan.

> **Note**
>
> Snapshot klaster DB harus dalam status `available` untuk dihapus.

Permintaan

- DBClusterSnapshotIdentifier(dalam CLI:`--db-cluster-snapshot-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi snapshot klaster DB yang akan dihapus.

  Kendala: Harus berupa nama snapshot klaster DB yang ada dalam status `available`.

Respons

Berisi detail untuk snapshot klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBCluster Snapshots”.

- AllocatedStorage— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan ukuran penyimpanan yang dialokasikan dalam gibibytes (GiB).
- AvailabilityZones— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance dalam snapshot cluster DB dapat dipulihkan.
- ClusterCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).
- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan pengidentifikasi klaster DB dari klaster DB tempat snapshot klaster DB ini dibuat.
- DBClusterSnapshotArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk snapshot klaster DB.

- DBClusterSnapshotIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan pengidentifikasi untuk snapshot klaster DB. Harus cocok dengan pengidentifikasi snapshot yang ada.

  Setelah Anda memulihkan klaster DB menggunakan `DBClusterSnapshotIdentifier`, Anda harus menentukan `DBClusterSnapshotIdentifier` yang sama untuk pembaruan klaster DB di masa mendatang. Ketika Anda menentukan properti ini untuk pembaruan, klaster DB tidak dipulihkan dari snapshot lagi, dan data dalam basis data tidak berubah.

  Namun, jika Anda tidak menentukan `DBClusterSnapshotIdentifier`, klaster DB kosong akan dibuat, dan klaster DB awal dihapus. Jika Anda menentukan properti yang berbeda dari properti pemulihan snapshot sebelumnya, klaster DB dipulihkan dari snapshot yang ditentukan oleh `DBClusterSnapshotIdentifier`, dan klaster DB awal dihapus.
- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama mesin basis data.
- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan versi mesin basis data untuk snapshot klaster DB ini.
- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.
- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah true, pengidentifikasi kunci Amazon KMS untuk snapshot klaster DB terenkripsi.
- LicenseModel— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan informasi model lisensi untuk snapshot klaster DB ini.
- PercentProgress— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan persentase perkiraan data yang telah ditransfer.
- Port— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan klaster DB pada waktu dari snapshot.

- SnapshotCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menyediakan waktu ketika snapshot diambil, dalam Universal Coordinated Time (UTC).
- SnapshotType— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan jenis snapshot klaster DB.
- SourceDBClusterSnapshotArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika snapshot klaster DB disalin dari snapshot klaster DB sumber, Amazon Resource Name (ARN) untuk snapshot klaster DB sumber; jika tidak, nilai null.
- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status snapshot klaster DB ini.
- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah snapshot klaster DB dienkripsi.
- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang terkait dengan snapshot cluster DB.
- VpcId— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan ID VPC terkait dengan snapshot klaster DB.

Kesalahan

- Tidak valid DBCluster SnapshotStateFault
- DBClusterSnapshotNotFoundFault

Salin DBCluster Snapshot (tindakan)

Nama AWS CLI untuk API ini adalah:. `copy-db-cluster-snapshot`

Menyalin snapshot dari klaster DB.

Untuk menyalin snapshot klaster DB dari snapshot klaster DB manual bersama, `SourceDBClusterSnapshotIdentifier` harus menjadi Amazon Resource Name (ARN) dari snapshot klaster DB bersama.

Permintaan

- CopyTags(dalam CLI:`--copy-tags`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  True untuk menyalin semua tag dari snapshot klaster DB sumber ke snapshot klaster DB target, dan sebaliknya false. Default-nya adalah false.

- KmsKeyId(dalam CLI:`--kms-key-id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ID kunci Amazon KMS untuk snapshot klaster DB terenkripsi. ID kunci KMS adalah Amazon Resource Name (ARN), pengidentifikasi kunci KMS, atau alias kunci KMS untuk kunci enkripsi KMS.

  Jika Anda menyalin snapshot klaster DB terenkripsi dari akun Amazon Anda, Anda dapat menentukan nilai untuk `KmsKeyId` untuk mengenkripsi salinan dengan kunci enkripsi KMS yang baru. Jika Anda tidak menentukan nilai untuk `KmsKeyId`, salinan snapshot klaster DB dienkripsi dengan kunci KMS yang sama dengan snapshot klaster DB sumber.

  Jika Anda menyalin snapshot klaster DB terenkripsi yang dibagikan dari aku Amazon lain, Anda harus menentukan nilai untuk `KmsKeyId`.

  Kunci enkripsi KMS dikhususkan untuk Wilayah Amazon yaitu tempat pembuatannya, dan Anda tidak dapat menggunakan kunci enkripsi dari satu Wilayah Amazon dalam Wilayah Amazon lainnya.

  Anda tidak dapat mengenkripsi snapshot klaster DB yang tidak dienkripsi saat Anda menyalinnya. Jika Anda mencoba menyalin snapshot cluster DB yang tidak terenkripsi dan menentukan nilai untuk KmsKeyId parameter, kesalahan akan dikembalikan.

- PreSignedUrl(dalam CLI:`--pre-signed-url`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Saat ini tidak didukung.

- SourceDBClusterSnapshotIdentifier(dalam CLI:`--source-db-cluster-snapshot-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi snapshot klaster DB yang akan disalin. Parameter ini tidak peka huruf besar kecil.

  Kendala:

  - Harus menentukan snapshot sistem yang valid dalam status “tersedia”.

- Menentukan pengidentifikasi snapshot DB yang valid.

  Contoh: `my-cluster-snapshot1`

- Tags(dalam CLI:`--tags`) — Sebuah array objek. Tag

  Tanda yang akan ditetapkan ke salinan snapshot klaster DB baru.

- TargetDBClusterSnapshotIdentifier(dalam CLI:`--target-db-cluster-snapshot-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi snapshot klaster DN baru yang akan dibuat dari snapshot klaster DB sumber. Parameter ini tidak peka huruf besar kecil.

  Kendala:

  - Harus berisi 1 sampai 63 huruf, angka, atau tanda hubung.
  - Karakter pertama harus berupa huruf.
  - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.

  Contoh: `my-cluster-snapshot2`

Respons

Berisi detail untuk snapshot klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBCluster Snapshots”.

- AllocatedStorage— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan ukuran penyimpanan yang dialokasikan dalam gibibytes (GiB).

- AvailabilityZones— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance dalam snapshot cluster DB dapat dipulihkan.

- ClusterCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).

- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

Menentukan pengidentifikasi klaster DB dari klaster DB tempat snapshot klaster DB ini dibuat.

- DBClusterSnapshotArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk snapshot klaster DB.

- DBClusterSnapshotIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan pengidentifikasi untuk snapshot klaster DB. Harus cocok dengan pengidentifikasi snapshot yang ada.

  Setelah Anda memulihkan klaster DB menggunakan `DBClusterSnapshotIdentifier`, Anda harus menentukan `DBClusterSnapshotIdentifier` yang sama untuk pembaruan klaster DB di masa mendatang. Ketika Anda menentukan properti ini untuk pembaruan, klaster DB tidak dipulihkan dari snapshot lagi, dan data dalam basis data tidak berubah.

  Namun, jika Anda tidak menentukan `DBClusterSnapshotIdentifier`, klaster DB kosong akan dibuat, dan klaster DB awal dihapus. Jika Anda menentukan properti yang berbeda dari properti pemulihan snapshot sebelumnya, klaster DB dipulihkan dari snapshot yang ditentukan oleh `DBClusterSnapshotIdentifier`, dan klaster DB awal dihapus.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama mesin basis data.

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan versi mesin basis data untuk snapshot klaster DB ini.

- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.

- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah true, pengidentifikasi kunci Amazon KMS untuk snapshot klaster DB terenkripsi.

- LicenseModel— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan informasi model lisensi untuk snapshot klaster DB ini.

- PercentProgress— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

Menentukan persentase perkiraan data yang telah ditransfer.

- Port— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan klaster DB pada waktu dari snapshot.

- SnapshotCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menyediakan waktu ketika snapshot diambil, dalam Universal Coordinated Time (UTC).

- SnapshotType— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan jenis snapshot klaster DB.

- SourceDBClusterSnapshotArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika snapshot klaster DB disalin dari snapshot klaster DB sumber, Amazon Resource Name (ARN) untuk snapshot klaster DB sumber; jika tidak, nilai null.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status snapshot klaster DB ini.

- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah snapshot klaster DB dienkripsi.

- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang terkait dengan snapshot cluster DB.

- VpcId— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan ID VPC terkait dengan snapshot klaster DB.

Kesalahan

- DBClusterSnapshotAlreadyExistsFault
- DBClusterSnapshotNotFoundFault
- Tidak valid DBCluster StateFault
- Tidak valid DBCluster SnapshotStateFault
- SnapshotQuotaExceededFault
- KMSKeyNotAccessibleFault

Memodifikasi DBCluster SnapshotAttribute (tindakan)

Nama AWS CLI untuk API ini adalah:. `modify-db-cluster-snapshot-attribute`

Menambahkan atribut dan nilai-nilai ke, atau menghapus atribut dan nilai-nilai dari, snapshot klaster DB manual.

Untuk membagikan snapshot cluster DB manual dengan akun Amazon lainnya, tentukan `restore` sebagai `AttributeName` dan gunakan `ValuesToAdd` parameter untuk menambahkan daftar IDs akun Amazon yang diizinkan untuk memulihkan snapshot cluster DB manual. Gunakan nilai `all` untuk membuat snapshot klaster DB manual menjadi publik, yang berarti bahwa itu dapat disalin atau dipulihkan oleh semua akun Amazon. Jangan menambahkan nilai `all` untuk setiap snapshot klaster DB manual yang berisi informasi privat yang Anda tidak ingin tersedia untuk semua akun Amazon. Jika snapshot cluster DB manual dienkripsi, snapshot tersebut dapat dibagikan, tetapi hanya dengan menentukan daftar akun IDs Amazon resmi untuk parameter tersebut. `ValuesToAdd` Anda tidak dapat menggunakan `all` sebagai nilai untuk parameter tersebut dalam kasus ini.

Untuk melihat akun Amazon mana yang memiliki akses untuk menyalin atau memulihkan snapshot klaster DB manual, atau apakah snapshot klaster DB manual publik atau privat, gunakan tindakan API the section called “Jelaskan DBCluster SnapshotAttributes”.

Permintaan

- AttributeName(dalam CLI:`--attribute-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama dari atribut snapshot klaster DB yang akan diubah.

  Untuk mengelola otorisasi bagi akun Amazon lain untuk menyalin atau memulihkan snapshot klaster DN manual, atur nilai ini ke `restore`.
- DBClusterSnapshotIdentifier(dalam CLI:`--db-cluster-snapshot-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi untuk snapshot klaster DB untuk memodifikasi atributnya.
- ValuesToAdd(dalam CLI:`--values-to-add`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar atribut snapshot klaster DB untuk ditambahkan ke atribut yang ditentukan oleh `AttributeName`.

Untuk mengotorisasi akun Amazon lainnya untuk menyalin atau memulihkan snapshot cluster DB manual, setel daftar ini untuk menyertakan satu atau beberapa akun Amazon IDs, atau `all` untuk membuat snapshot cluster DB manual dapat dipulihkan oleh akun Amazon mana pun. Jangan menambahkan nilai `all` untuk setiap snapshot klaster DB manual yang berisi informasi privat yang Anda tidak ingin tersedia untuk semua akun Amazon.

- ValuesToRemove(dalam CLI:`--values-to-remove`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar atribut snapshot klaster DB untuk dihapus dari atribut yang ditentukan oleh `AttributeName`.

  Untuk menghapus otorisasi untuk akun Amazon lain untuk menyalin atau memulihkan snapshot klaster DB manual, gatur daftar ini untuk menyertakan satu atau beberapa pengidentifikasi akun Amazon, atau `all` untuk menghapus otorisasi untuk akun Amazon mana pun untuk menyalin atau memulihkan snapshot klaster DB. Jika Anda menentukan `all`, sebuah akun Amazon yang ID akunnya secara eksplisit ditambahkan ke atribut `restore` masih dapat menyalin atau memulihkan snapshot klaster DB manual.

Respons

Berisi hasil dari panggilan sukses ke tindakan API the section called “Jelaskan DBCluster SnapshotAttributes”.

Atribut snapshot klaster DB manual digunakan untuk mengotorisasi akun Amazon lain untuk memulihkan snapshot klaster DB manual. Untuk informasi selengkapnya, lihat tindakan API the section called “Memodifikasi DBCluster SnapshotAttribute”.

- DBClusterSnapshotAttributes – Susunan objek DBClusterSnapshotAttribute.

  Daftar atribut dan nilai untuk snapshot klaster DB manual.

- DBClusterSnapshotIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi dari snapshot klaster DB manual tempat atribut diterapkan.

Kesalahan

- DBClusterSnapshotNotFoundFault

- Tidak valid DBCluster SnapshotStateFault
- SharedSnapshotQuotaExceededFault

Kembalikan DBCluster FromSnapshot (tindakan)

Nama AWS CLI untuk API ini adalah:. `restore-db-cluster-from-snapshot`

Membuat klaster DB baru dari snapshot DB atau snapshot klaster DB.

Jika snapshot ditentukan DB, klaster DB target dibuat dari snapshot DB sumber dengan konfigurasi default dan grup keamanan default.

Jika snapshot klaster DB ditentukan, klaster target DB dibuat dari titik pemulihan klaster DB sumber dengan konfigurasi yang sama seperti klaster DB sumber asli, kecuali klaster DB baru dibuat dengan grup keamanan default.

Permintaan

- AvailabilityZones(dalam CLI:`--availability-zones`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance di cluster DB yang dipulihkan dapat dibuat.
- CopyTagsToSnapshot(dalam CLI:`--copy-tags-to-snapshot`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke*`true`*, tag akan disalin ke snapshot apa pun dari cluster DB yang dipulihkan yang dibuat.
- DatabaseName(dalam CLI:`--database-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung.
- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama klaster DB yang akan dibuat dari snapshot DB atau snapshot klaster DB. Parameter ini tidak peka huruf besar-kecil.

  Kendala:

- Harus berisi 1 sampai 63 huruf, angka, atau tanda hubung
- Karakter pertamanya harus berupa huruf
- Tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung yang berurutan

Contoh: `my-snapshot-id`

- DBClusterParameterGroupName(dalam CLI:`--db-cluster-parameter-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter klaster DB yang akan dikaitkan dengan klaster DB ini.

  Batasan:

  - Jika disediakan, harus cocok dengan nama yang sudah ada DBClusterParameterGroup.
- DBSubnetGroupName(dalam CLI:`--db-subnet-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup subnet DB yang akan digunakan untuk klaster DB baru.

  Kendala: Jika disediakan, harus cocok dengan nama Grup yang ada. DBSubnet

  Contoh: `mySubnetgroup`
- DeletionProtection(dalam CLI:`--deletion-protection`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang menunjukkan apakah klaster DB memiliki perlindungan penghapusan yang diaktifkan. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan. Secara default, perlindungan penghapusan dinonaktifkan.
- EnableCloudwatchLogsExports(dalam CLI:`--enable-cloudwatch-logs-exports`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar log yang akan diekspor oleh cluster DB yang dipulihkan ke Amazon CloudWatch Logs.
- EnableIAMDatabaseAuthentication(dalam CLI:`--enable-iam-database-authentication`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  True untuk mengaktifkan pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data, dan sebaliknya false.

  Default: `false`
- Engine(dalam CLI:`--engine`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

Mesin basis data yang akan digunakan untuk klaster DB baru.

Default: Sama seperti sumber

Kendala: Harus kompatibel dengan mesin sumber.

- EngineVersion(dalam CLI:`--engine-version`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Versi mesin basis data yang akan digunakan untuk klaster DB baru.

- KmsKeyId(dalam CLI:`--kms-key-id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi kunci Amazon KMS untuk digunakan saat memulihkan klaster DB terenkripsi dari snapshot DB atau snapshot klaster DB.

  Pengidentifikasi kunci KMS adalah Amazon Resource Name (ARN) untuk kunci enkripsi KMS. Jika Anda memulihkan klaster DB dengan akun Amazon yang sama yang memiliki kunci enkripsi KMS yang digunakan untuk mengenkripsi klaster DB baru, maka Anda dapat menggunakan alias kunci KMS alih-alih ARN untuk kunci enkripsi KMS.

  Jika Anda tidak menentukan nilai untuk parameter `KmsKeyId`, maka hal berikut akan terjadi:

  - Jika snapshot DB atau snapshot klaster DB di `SnapshotIdentifier` dienkripsi, maka klaster DB yang dipulihkan dienkripsi menggunakan kunci KMS yang digunakan untuk mengenkripsi snapshot DB atau snapshot klaster DB.
  - Jika snapshot DB atau snapshot klaster DB di `SnapshotIdentifier` tidak dienkripsi, maka klaster DB yang dipulihkan tidak dienkripsi.

- Port(dalam CLI:`--port`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Nomor port tempat klaster DB baru menerima koneksi.

  Kendala: Nilai harus `1150-65535`

  Default: Port yang sama dengan klaster DB asli.

- ServerlessV2ScalingConfiguration(dalam CLI:`--serverless-v2-scaling-configuration`) — Sebuah ServerlessV2 ScalingConfiguration objek.

  Berisi konfigurasi penskalaan cluster DB Neptunus Tanpa Server.

Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- SnapshotIdentifier(dalam CLI:`--snapshot-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi snapshot DB atau snapshot klaster DB yang akan dipulihkan.

  Anda dapat menggunakan nama atau Amazon Resource Name (ARN) untuk menentukan snapshot klaster DB. Namun, Anda dapat menggunakan ARN saja untuk menentukan snapshot DB.

  Kendala:

  - Harus cocok dengan pengidentifikasi snapshot yang ada.
- StorageType(dalam CLI:`--storage-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan jenis penyimpanan yang akan dikaitkan dengan cluster DB.

  Nilai valid: `standard`, `iopt1`

  Default: `standard`
- Tags(dalam CLI:`--tags`) — Sebuah array objek. Tag

  Tanda yang akan ditetapkan ke klaster Db yang dipulihkan.
- VpcSecurityGroupIds(dalam CLI:`--vpc-security-group-ids`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar grup keamanan VPC tempat klaster DB baru akan berada.

Respons

Berisi detail dari klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam filethe section called “Jelaskan DBClusters”.

- AllocatedStorage— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  `AllocatedStorage` selalu mengembalikan 1, karena ukuran penyimpanan klaster DB Neptune tidak tetap, tetapi justru secara otomatis menyesuaikan sesuai kebutuhan.

- AssociatedRoles – Susunan objek DBClusterPeran.

  Menyediakan daftar Amazon Identity and Access Management (IAM) role yang terkait dengan klaster DB. IAM role yang terkait dengan klaster DB memberikan izin pada klaster DB untuk mengakses layanan Amazon lain atas nama Anda.
- AutomaticRestartTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Waktu di mana klaster DB akan secara otomatis di-restart.
- AvailabilityZones— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance di cluster DB dapat dibuat.
- BacktrackConsumedChangeRecords- a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.
- BacktrackWindow- a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.
- BackupRetentionPeriod— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.
- Capacity— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Tidak didukung oleh Neptune.
- CloneGroupId— String, tipe: `string` (string yang dikodekan UTF-8).

  Mengidentifikasi grup klon yang terkait dengan klaster DB.
- ClusterCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).
- CopyTagsToSnapshot— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke*`true`*, tag akan disalin ke snapshot apa pun dari cluster DB yang dibuat.
- CrossAccountClone— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika diatur ke `true`, klaster DB dapat dikloning di seluruh akun.

- DatabaseName— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama basis data awal klaster DB ini yang disediakan pada waktu membuat, jika salah satu ditentukan ketika klaster DB dibuat. Nama yang sama ini dikembalikan demi kluster DB.
- DBClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk klaster DB.
- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengidentifikasi klaster DB yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi klaster DB.
- DBClusterMembers – Susunan objek DBClusterAnggota.

  Menyediakan daftar instans yang membentuk klaster DB.
- DBClusterParameterGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama grup parameter klaster DB untuk klaster DB ini.
- DbClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi Wilayah Amazon unik yang tetap untuk klaster. Pengenal ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk cluster DB diakses.
- DBSubnetGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan informasi pada grup subnet yang terkait dengan klaster DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.
- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan.
- EarliestBacktrackTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tidak didukung oleh Neptune.
- EarliestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu paling awal dimana database dapat dipulihkan dengan point-in-time restore.
- EnabledCloudwatchLogsExports— String, tipe: `string` (string yang dikodekan UTF-8).

Daftar jenis log yang cluster DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log. Jenis log yang valid adalah: `audit` (untuk mempublikasikan log audit ke CloudWatch) dan slowquery (untuk mempublikasikan log kueri lambat ke). CloudWatch Lihat Menerbitkan log Neptunus ke log Amazon. CloudWatch

- Endpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan titik akhir koneksi untuk instans primer klaster DB.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk klaster DB ini.

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.

- HostedZoneId— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan ID yang ditetapkan Amazon Route 53 saat Anda membuat zona yang di-hosting.

- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.

- IOOptimizedNextAllowedModificationTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Lain kali Anda dapat memodifikasi cluster DB untuk menggunakan jenis `iopt1` penyimpanan.

- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah BETUL, pengidentifikasi kunci Amazon KMS untuk klaster DB terenkripsi.

- LatestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.

- MultiAZ— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB memiliki instans di beberapa Availability Zone.

- PendingModifiedValues — Sebuah objek ClusterPendingModifiedValues.

  Tipe data ini digunakan sebagai elemen respons dalam `ModifyDBCluster` operasi dan berisi perubahan yang akan diterapkan selama jendela pemeliharaan berikutnya.

- PercentProgress— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan kemajuan operasi sebagai persentase.

- Port— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan oleh mesin basis data.

- PreferredBackupWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.

- PreferredMaintenanceWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).

- ReaderEndpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Reader Endpoint untuk klaster DB. Reader Endpoint untuk koneksi menyeimbangkan beban klaster DB di seluruh Replika Baca yang tersedia dalam klaster DB. Saat klien meminta koneksi baru ke Reader Endpoint, Neptune mendistribusikan permintaan koneksi di antara Replika Neptune di klaster DB. Fungsionalitas ini dapat membantu menyeimbangkan beban kerja baca Anda di beberapa Replika Baca di klaster DB Anda.

  Jika terjadi failover, dan Replika Baca yang terhubung dengan Anda dipromosikan menjadi instans primer, koneksi Anda dijatuhkan. Untuk terus mengirimkan beban kerja baca Anda ke Replika Baca lainnya di klaster, Anda kemudian dapat menyambungkan kembali ke Reader Endpoint.

- ReadReplicaIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi Replika Baca terkait dengan klaster DB ini.

- ReplicationSourceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

Tidak didukung oleh Neptune.

- ReplicationType— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ServerlessV2ScalingConfiguration — Sebuah objek ServerlessV2 ScalingConfigurationInfo.

  Menampilkan konfigurasi penskalaan untuk cluster DB Neptunus Tanpa Server.

  Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari klaster DB ini.

- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB dienkripsi.

- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang digunakan oleh cluster DB.

  Nilai yang Valid:

  - **`standard`**— (default) Menyediakan penyimpanan database hemat biaya untuk aplikasi dengan penggunaan sedang hingga kecil I/O .
  - **`iopt1`**— Memungkinkan penyimpanan I/O-Optimized yang dirancang untuk memenuhi kebutuhan throughput. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    Penyimpanan Neptunus I/O-Optimized hanya tersedia dimulai dengan rilis engine 1.3.0.0.

- VpcSecurityGroups – Susunan objek VpcSecurityGroupMembership.

  Memberikan daftar grup keamanan VPC yang memiliki klaster DB.

Kesalahan

- DBClusterAlreadyExistsFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault

- DBSubnetGroupNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterSnapshotNotFoundFault
- Tidak cukup DBCluster CapacityFault
- InsufficientStorageClusterCapacityFault
- Tidak valid DBSnapshot StateFault
- Tidak valid DBCluster SnapshotStateFault
- StorageQuotaExceededFault
- Tidak valid VPCNetwork StateFault
- InvalidRestoreFault
- DBSubnetGroupNotFoundFault
- InvalidSubnet
- OptionGroupNotFoundFault
- KMSKeyNotAccessibleFault
- DBClusterParameterGroupNotFoundFault

Kembalikan DBCluster ToPointInTime (tindakan)

Nama AWS CLI untuk API ini adalah:. `restore-db-cluster-to-point-in-time`

Memulihkan klaster DB ke titik waktu yang arbitrer. Pengguna dapat memulihkan ke titik waktu mana pun sebelum `LatestRestorableTime` hingga `BackupRetentionPeriod` hari. Klaster DB target dibuat dari klaster Db sumber dengan konfigurasi yang sama seperti klaster Db asli, kecuali klaster DB baru dibuat dengan grup keamanan DB default.

> **Note**
>
> Tindakan ini memulihkan hanya klaster DB, dan bukan instans Dbuntuk klaster DB tersebut. Anda harus memanggil tindakan the section called “Buat DBInstance” guna membuat instans DB untuk klaster DB yang dipulihkan, menentukan pengidentifikasi klaster DB yang dipulihkan di `DBClusterIdentifier`. Anda dapat membuat instans DB hanya setelah tindakan `RestoreDBClusterToPointInTime` telah selesai dan klaster DB tersedia.

Permintaan

- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama klaster DB baru yang akan dibuat.

  Kendala:

  - Harus berisi 1 sampai 63 huruf, angka, atau tanda hubung
  - Karakter pertamanya harus berupa huruf
  - Tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung yang berurutan
- DBClusterParameterGroupName(dalam CLI:`--db-cluster-parameter-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup parameter klaster DB yang akan dikaitkan dengan klaster DB ini.

  Batasan:

  - Jika disediakan, harus cocok dengan nama yang sudah ada DBClusterParameterGroup.
- DBSubnetGroupName(dalam CLI:`--db-subnet-group-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup subnet DB yang akan digunakan untuk klaster DB baru.

  Kendala: Jika disediakan, harus cocok dengan nama Grup yang ada. DBSubnet

  Contoh: `mySubnetgroup`
- DeletionProtection(dalam CLI:`--deletion-protection`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang menunjukkan apakah klaster DB memiliki perlindungan penghapusan yang diaktifkan. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan. Secara default, perlindungan penghapusan dinonaktifkan.
- EnableCloudwatchLogsExports(dalam CLI:`--enable-cloudwatch-logs-exports`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar log yang akan diekspor oleh cluster DB yang dipulihkan ke CloudWatch Log.
- EnableIAMDatabaseAuthentication(dalam CLI:`--enable-iam-database-authentication`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

True untuk mengaktifkan pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data, dan sebaliknya false.

Default: `false`

- KmsKeyId(dalam CLI:`--kms-key-id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi kunci Amazon KMS untuk digunakan saat memulihkan klaster DB terenkripsi dari kluster DB terenkripsi.

  Pengidentifikasi kunci KMS adalah Amazon Resource Name (ARN) untuk kunci enkripsi KMS. Jika Anda memulihkan klaster DB dengan akun Amazon yang sama yang memiliki kunci enkripsi KMS yang digunakan untuk mengenkripsi klaster DB baru, maka Anda dapat menggunakan alias kunci KMS alih-alih ARN untuk kunci enkripsi KMS.

  Anda dapat memulihkan ke klaster DB baru dan mengenkripsi klaster DB baru dengan kunci KMS yang berbeda dari kunci KMS yang digunakan untuk mengenkripsi klaster DB sumber. Klaster DB baru dienkripsi dengan kunci KMS yang diidentifikasi oleh parameter `KmsKeyId`.

  Jika Anda tidak menentukan nilai untuk parameter `KmsKeyId`, maka hal berikut akan terjadi:

  - Jika klaster DB dienkripsi, maka klaster DB yang dipulihkan dienkripsi menggunakan kunci KMS yang digunakan untuk mengenkripsi klaster DB sumber.
  - Jika klaster DB tidak dienkripsi, maka klaster DB yang dipulihkan tidak dienkripsi.

  Jika `DBClusterIdentifier` mengacu pada sebuah klaster DB yang tidak dienkripsi, maka permintaan pemulihan ditolak.
- Port(dalam CLI:`--port`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Nomor port tempat klaster DB baru menerima koneksi.

  Kendala: Nilai harus `1150-65535`

  Default: Port yang sama dengan klaster DB asli.
- RestoreToTime(dalam CLI:`--restore-to-time`) — a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tanggal dan waktu untuk memulihkan klaster DB.

Nilai yang Valid: Nilai harus berupa waktu dalam format Waktu Universal Terkoordinasi (UTC).

Kendala:

- Harus sebelum waktu pemulihan terbaru untuk instans DB.
- Harus ditentukan jika parameter `UseLatestRestorableTime` tidak disediakan.
- Tidak dapat ditentukan jika parameter `UseLatestRestorableTime` adalah true.
- Tidak dapat ditentukan jika parameter `RestoreType` adalah `copy-on-write`.

Contoh: `2015-03-07T23:45:00Z`

- RestoreType(dalam CLI:`--restore-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Tipe pemulihan yang akan dilakukan. Anda dapat menentukan salah satu nilai berikut:

  - `full-copy` - Klaster DB baru dipulihkan sebagai salinan lengkap dari klaster DB sumber.
  - `copy-on-write` - Klaster DB baru dipulihkan sebagai klon dari klaster DB sumber.

  Jika Anda tidak menentukan nilai `RestoreType`, maka klaster DB baru dipulihkan sebagai salinan lengkap klaster DB sumber.

- ServerlessV2ScalingConfiguration(dalam CLI:`--serverless-v2-scaling-configuration`) — Sebuah ServerlessV2 ScalingConfiguration objek.

  Berisi konfigurasi penskalaan cluster DB Neptunus Tanpa Server.

  Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- SourceDBClusterIdentifier(dalam CLI:`--source-db-cluster-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi klaster DB sumber yang akan dipulihkan.

  Batasan:

  - Harus cocok dengan pengenal yang sudah ada DBCluster.

- StorageType(dalam CLI:`--storage-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan jenis penyimpanan yang akan dikaitkan dengan cluster DB.

Nilai valid: `standard`, `iopt1`

Default: `standard`

- Tags(dalam CLI:`--tags`) — Sebuah array objek. Tag

  Tanda yang akan diterapkan ke klaster Db yang dipulihkan.
- UseLatestRestorableTime(dalam CLI:`--use-latest-restorable-time`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang diatur ke `true` untuk memulihkan klaster DB ke waktu backup terbaru yang dapat dipulihkan, dan `false` sebaliknya.

  Default: `false`

  Kendala: Tidak dapat ditentukan jika parameter `RestoreToTime` disediakan.
- VpcSecurityGroupIds(dalam CLI:`--vpc-security-group-ids`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar grup keamanan VPC tempat klaster DB baru berada.

Respons

Berisi detail dari klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam filethe section called “Jelaskan DBClusters”.

- AllocatedStorage— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  `AllocatedStorage` selalu mengembalikan 1, karena ukuran penyimpanan klaster DB Neptune tidak tetap, tetapi justru secara otomatis menyesuaikan sesuai kebutuhan.
- AssociatedRoles – Susunan objek DBClusterPeran.

  Menyediakan daftar Amazon Identity and Access Management (IAM) role yang terkait dengan klaster DB. IAM role yang terkait dengan klaster DB memberikan izin pada klaster DB untuk mengakses layanan Amazon lain atas nama Anda.
- AutomaticRestartTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Waktu di mana klaster DB akan secara otomatis di-restart.

- AvailabilityZones— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance di cluster DB dapat dibuat.

- BacktrackConsumedChangeRecords- a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- BacktrackWindow- a LongOptional, tipe: `long` (integer 64-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- BackupRetentionPeriod— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan jumlah hari yang mana snapshot DB otomatis dipertahankan.

- Capacity— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Tidak didukung oleh Neptune.

- CloneGroupId— String, tipe: `string` (string yang dikodekan UTF-8).

  Mengidentifikasi grup klon yang terkait dengan klaster DB.

- ClusterCreateTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).

- CopyTagsToSnapshot— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke*`true`*, tag akan disalin ke snapshot apa pun dari cluster DB yang dibuat.

- CrossAccountClone— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika diatur ke `true`, klaster DB dapat dikloning di seluruh akun.

- DatabaseName— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi nama basis data awal klaster DB ini yang disediakan pada waktu membuat, jika salah satu ditentukan ketika klaster DB dibuat. Nama yang sama ini dikembalikan demi kluster DB.

- DBClusterArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk klaster DB.

- DBClusterIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

Berisi pengidentifikasi klaster DB yang disediakan pengguna. Pengidentifikasi ini adalah kunci unik yang mengidentifikasi klaster DB.

- DBClusterMembers – Susunan objek DBClusterAnggota.

  Menyediakan daftar instans yang membentuk klaster DB.

- DBClusterParameterGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama grup parameter klaster DB untuk klaster DB ini.

- DbClusterResourceId— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi Wilayah Amazon unik yang tetap untuk klaster. Pengenal ini ditemukan di entri CloudTrail log Amazon setiap kali kunci Amazon KMS untuk cluster DB diakses.

- DBSubnetGroup— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan informasi pada grup subnet yang terkait dengan klaster DB, termasuk nama, deskripsi, dan subnet dalam grup subnet.

- DeletionProtection— a BooleanOptional, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah klaster DB memiliki perlindungan penghapusan diaktifkan. Basis data global tidak dapat dihapus saat perlindungan penghapusan diaktifkan.

- EarliestBacktrackTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tidak didukung oleh Neptune.

- EarliestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu paling awal dimana database dapat dipulihkan dengan point-in-time restore.

- EnabledCloudwatchLogsExports— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar jenis log yang cluster DB ini dikonfigurasi untuk mengekspor ke CloudWatch Log. Jenis log yang valid adalah: `audit` (untuk mempublikasikan log audit ke CloudWatch) dan slowquery (untuk mempublikasikan log kueri lambat ke). CloudWatch Lihat Menerbitkan log Neptunus ke log Amazon. CloudWatch

- Endpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan titik akhir koneksi untuk instans primer klaster DB.

- Engine— String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan nama mesin basis data yang akan digunakan untuk klaster DB ini.

- EngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan versi mesin basis data.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, tipe: `string` (string yang dikodekan UTF-8), tidak kurang dari 1 atau lebih dari 255? st? s, cocok dengan ekspresi reguler ini:`[A-Za-z][0-9A-Za-z-:._]*`.

  Berisi pengidentifikasi cluster database global yang disediakan pengguna. Identifier ini adalah kunci unik yang mengidentifikasi database global.

- HostedZoneId— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan ID yang ditetapkan Amazon Route 53 saat Anda membuat zona yang di-hosting.

- IAMDatabaseAuthenticationEnabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.

- IOOptimizedNextAllowedModificationTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Lain kali Anda dapat memodifikasi cluster DB untuk menggunakan jenis `iopt1` penyimpanan.

- KmsKeyId— String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah BETUL, pengidentifikasi kunci Amazon KMS untuk klaster DB terenkripsi.

- LatestRestorableTime— a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu terbaru yang database dapat dipulihkan dengan point-in-time restore.

- MultiAZ— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB memiliki instans di beberapa Availability Zone.

- PendingModifiedValues — Sebuah objek ClusterPendingModifiedValues.

  Tipe data ini digunakan sebagai elemen respons dalam `ModifyDBCluster` operasi dan berisi perubahan yang akan diterapkan selama jendela pemeliharaan berikutnya.

- PercentProgress— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan kemajuan operasi sebagai persentase.

- Port— an IntegerOptional, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan oleh mesin basis data.

- PreferredBackupWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu harian selama backup otomatis dibuat jika backup otomatis diaktifkan, seperti yang ditentukan oleh `BackupRetentionPeriod`.

- PreferredMaintenanceWindow— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan rentang waktu mingguan selama pemeliharaan sistem dapat dilakukan, dalam Waktu Universal Terkoordinasi (UTC).

- ReaderEndpoint— String, tipe: `string` (string yang dikodekan UTF-8).

  Reader Endpoint untuk klaster DB. Reader Endpoint untuk koneksi menyeimbangkan beban klaster DB di seluruh Replika Baca yang tersedia dalam klaster DB. Saat klien meminta koneksi baru ke Reader Endpoint, Neptune mendistribusikan permintaan koneksi di antara Replika Neptune di klaster DB. Fungsionalitas ini dapat membantu menyeimbangkan beban kerja baca Anda di beberapa Replika Baca di klaster DB Anda.

  Jika terjadi failover, dan Replika Baca yang terhubung dengan Anda dipromosikan menjadi instans primer, koneksi Anda dijatuhkan. Untuk terus mengirimkan beban kerja baca Anda ke Replika Baca lainnya di klaster, Anda kemudian dapat menyambungkan kembali ke Reader Endpoint.

- ReadReplicaIdentifiers— String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi satu atau lebih pengidentifikasi Replika Baca terkait dengan klaster DB ini.

- ReplicationSourceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ReplicationType— String, tipe: `string` (string yang dikodekan UTF-8).

  Tidak didukung oleh Neptune.

- ServerlessV2ScalingConfiguration — Sebuah objek ServerlessV2 ScalingConfigurationInfo.

  Menampilkan konfigurasi penskalaan untuk cluster DB Neptunus Tanpa Server.

Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status saat ini dari klaster DB ini.

- StorageEncrypted— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah klaster DB dienkripsi.

- StorageType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang digunakan oleh cluster DB.

  Nilai yang Valid:

  - **`standard`**— (default) Menyediakan penyimpanan database hemat biaya untuk aplikasi dengan penggunaan sedang hingga kecil I/O .
  - **`iopt1`**— Memungkinkan penyimpanan I/O-Optimized yang dirancang untuk memenuhi kebutuhan throughput. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    Penyimpanan Neptunus I/O-Optimized hanya tersedia dimulai dengan rilis engine 1.3.0.0.

- VpcSecurityGroups – Susunan objek VpcSecurityGroupMembership.

  Memberikan daftar grup keamanan VPC yang memiliki klaster DB.

Kesalahan

- DBClusterAlreadyExistsFault
- DBClusterNotFoundFault
- DBClusterQuotaExceededFault
- DBClusterSnapshotNotFoundFault
- DBSubnetGroupNotFoundFault
- Tidak cukup DBCluster CapacityFault
- InsufficientStorageClusterCapacityFault
- Tidak valid DBCluster SnapshotStateFault
- Tidak valid DBCluster StateFault

- Tidak valid DBSnapshot StateFault
- InvalidRestoreFault
- InvalidSubnet
- Tidak valid VPCNetwork StateFault
- KMSKeyNotAccessibleFault
- OptionGroupNotFoundFault
- StorageQuotaExceededFault
- DBClusterParameterGroupNotFoundFault

Jelaskan DBCluster Snapshots (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-db-cluster-snapshots`

Mengembalikan informasi tentang snapshot klaster DB. Tindakan API ini mendukung pagination (pemberian nomor halaman).

Permintaan

- DBClusterIdentifier(dalam CLI:`--db-cluster-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ID dari klaster DB untuk mengambil daftar snapshot klaster DB. Parameter ini tidak dapat digunakan bersamaan dengan parameter `DBClusterSnapshotIdentifier`. Parameter ini tidak peka huruf besar kecil.

  Batasan:

  - Jika disediakan, harus cocok dengan pengenal yang sudah ada DBCluster.
- DBClusterSnapshotIdentifier(dalam CLI:`--db-cluster-snapshot-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi snapshot klaster DB tertentu untuk menggambarkan. Parameter ini tidak dapat digunakan bersamaan dengan parameter `DBClusterIdentifier`. Nilai ini disimpan sebagai string huruf kecil.

  Batasan:

  - Jika disediakan, harus cocok dengan pengenal DBCluster Snapshot yang ada.

  - Jika pengidentifikasi ini adalah untuk snapshot otomatis, parameter `SnapshotType` juga harus ditentukan.
- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Parameter ini saat ini tidak didukung.
- IncludePublic(dalam CLI:`--include-public`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  True untuk menyertakan snapshot klaster DB manual yang bersifat publik dan dapat disalin atau dipulihkan oleh akun Amazon mana pun, dan sebaliknya false. Nilai default-nya `false`. Default-nya adalah false.

  Anda dapat berbagi snapshot klaster DB manual sebagai publik dengan menggunakan tindakan API the section called “Memodifikasi DBCluster SnapshotAttribute”.
- IncludeShared(dalam CLI:`--include-shared`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  True untuk menyertakan snapshot klaster DB manual bersama dari akun Amazon lain yang akun Amazon ini telah diberikan izin untuk menyalin atau memulihkan, dan sebaliknya false. Nilai default-nya `false`.

  Anda dapat memberikan izin akun Amazon untuk memulihkan snapshot klaster DB manual dari akun Amazon lain dengan tindakan API the section called “Memodifikasi DBCluster SnapshotAttribute”.
- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan `DescribeDBClusterSnapshots` sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.
- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

  Default: 100

Kendala: Minimal 20, maksimum 100.

- SnapshotType(dalam CLI:`--snapshot-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis snapshot klaster Db untuk dikembalikan. Anda dapat menentukan salah satu nilai berikut:

  - `automated` - Mengembalikan semua snapshot klaster DB yang telah secara otomatis diambil oleh Amazon Neptune untuk akun Amazon saya.
  - `manual` - Mengembalikan semua snapshot klaster DB yang telah diambil oleh akun Amazon saya.
  - `shared` - Mengembalikan semua snapshot klaster manual yang telah dibagikan ke akun Amazon saya.
  - `public` - Mengembalikan semua snapshot klaster DB yang telah ditandai sebagai publik.

  Jika Anda tidak menentukan nilai `SnapshotType`, maka baik snapshot klaster DB otomatis dan manual dikembalikan. Anda dapat menyertakan snapshot klaster DB bersama dengan hasil ini dengan mengatur parameter `IncludeShared` ke `true`. Anda dapat menyertakan snapshot klaster DB publik dengan hasil ini dengan mengatur parameter `IncludePublic` ke `true`.

  Parameter `IncludeShared` dan `IncludePublic` tidak berlaku untuk nilai `SnapshotType` dari `manual` atau `automated`. Parameter `IncludePublic` tidak berlaku bila `SnapshotType` diatur ke `shared`. Parameter `IncludeShared` tidak berlaku bila `SnapshotType` diatur ke `public`.

Respons

- DBClusterSnapshots – Susunan objek DBClusterCuplikan.

  Menyediakan daftar snapshot klaster DB untuk pengguna.

- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan the section called “Jelaskan DBCluster Snapshots” sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

Kesalahan

- DBClusterSnapshotNotFoundFault

Jelaskan DBCluster SnapshotAttributes (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-db-cluster-snapshot-attributes`

Mengembalikan daftar nama atribut dan nilao snapshot klaster DB untuk snapshot klaster DB manual.

Saat berbagi snapshot dengan akun Amazon lainnya, `DescribeDBClusterSnapshotAttributes` mengembalikan `restore` atribut dan daftar IDs untuk akun Amazon yang diizinkan untuk menyalin atau memulihkan snapshot cluster DB manual. Jika `all` disertakan dalam daftar nilai untuk atribut `restore`, maka snapshot klaster DB manual bersifat publik dan dapat disalin atau dipulihkan oleh semua akun Amazon.

Untuk menambah atau menghapus akses untuk akun Amazon untuk menyalin atau memulihkan snapshot klaster DB manual, atau untuk membuat snapshot klaster DB publik atau privat, gunakan tindakan API the section called “Memodifikasi DBCluster SnapshotAttribute”.

Permintaan

- DBClusterSnapshotIdentifier(dalam CLI:`--db-cluster-snapshot-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi untuk snapshot klaster DB untuk mendeskripsikan atributnya.

Respons

Berisi hasil dari panggilan sukses ke tindakan API the section called “Jelaskan DBCluster SnapshotAttributes”.

Atribut snapshot klaster DB manual digunakan untuk mengotorisasi akun Amazon lain untuk memulihkan snapshot klaster DB manual. Untuk informasi selengkapnya, lihat tindakan API the section called “Memodifikasi DBCluster SnapshotAttribute”.

- DBClusterSnapshotAttributes – Susunan objek DBClusterSnapshotAttribute.

  Daftar atribut dan nilai untuk snapshot klaster DB manual.
- DBClusterSnapshotIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi dari snapshot klaster DB manual tempat atribut diterapkan.

Kesalahan

- DBClusterSnapshotNotFoundFault

Struktur:

DBClusterSnapshot (struktur)

Berisi detail untuk snapshot klaster DB Amazon Neptune.

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBCluster Snapshots”.

Bidang

- AllocatedStorage— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan ukuran penyimpanan yang dialokasikan dalam gibibytes (GiB).
- AvailabilityZones— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan daftar EC2 Availability Zones dimana instance dalam snapshot cluster DB dapat dipulihkan.
- ClusterCreateTime— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menentukan waktu saat klaster DB dibuat, dalam Waktu Universal Terkoordinasi (UTC).
- DBClusterIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan pengidentifikasi klaster DB dari klaster DB tempat snapshot klaster DB ini dibuat.
- DBClusterSnapshotArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk snapshot klaster DB.
- DBClusterSnapshotIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan pengidentifikasi untuk snapshot klaster DB. Harus cocok dengan pengidentifikasi snapshot yang ada.

  Setelah Anda memulihkan klaster DB menggunakan `DBClusterSnapshotIdentifier`, Anda harus menentukan `DBClusterSnapshotIdentifier` yang sama untuk pembaruan klaster

DB di masa mendatang. Ketika Anda menentukan properti ini untuk pembaruan, klaster DB tidak dipulihkan dari snapshot lagi, dan data dalam basis data tidak berubah.

Namun, jika Anda tidak menentukan `DBClusterSnapshotIdentifier`, klaster DB kosong akan dibuat, dan klaster DB awal dihapus. Jika Anda menentukan properti yang berbeda dari properti pemulihan snapshot sebelumnya, klaster DB dipulihkan dari snapshot yang ditentukan oleh `DBClusterSnapshotIdentifier`, dan klaster DB awal dihapus.

- Engine— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama mesin basis data.

- EngineVersion— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan versi mesin basis data untuk snapshot klaster DB ini.

- IAMDatabaseAuthenticationEnabled— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  BETUL jika pemetaan akun Amazon Identity and Access Management (IAM) ke akun basis data akun diaktifkan, dan sebaliknya SALAH.

- KmsKeyId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jika `StorageEncrypted` adalah true, pengidentifikasi kunci Amazon KMS untuk snapshot klaster DB terenkripsi.

- LicenseModel— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan informasi model lisensi untuk snapshot klaster DB ini.

- PercentProgress— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan persentase perkiraan data yang telah ditransfer.

- Port— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan klaster DB pada waktu dari snapshot.

- SnapshotCreateTime— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Menyediakan waktu ketika snapshot diambil, dalam Universal Coordinated Time (UTC).

- SnapshotType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan jenis snapshot klaster DB.

- SourceDBClusterSnapshotArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jika snapshot klaster DB disalin dari snapshot klaster DB sumber, Amazon Resource Name (ARN) untuk snapshot klaster DB sumber; jika tidak, nilai null.
- Status— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan status snapshot klaster DB ini.
- StorageEncrypted— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menentukan apakah snapshot klaster DB dienkripsi.
- StorageType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis penyimpanan yang terkait dengan snapshot cluster DB.
- VpcId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan ID VPC terkait dengan snapshot klaster DB.

`DBClusterSnapshot` digunakan sebagai elemen respons untuk:

- Buat DBCluster Snapshot
- Salin DBCluster Snapshot
- Hapus DBCluster Snapshot

DBClusterSnapshotAttribute (struktur)

Berisi nama dan nilai-nilai atribut snapshot klaster DB manual.

Atribut snapshot klaster Db manual digunakan untuk mengotorisasi akun Amazon lain untuk memulihkan snapshot klaster DB manual. Untuk informasi selengkapnya, lihat tindakan API the section called “Memodifikasi DBCluster SnapshotAttribute”.

Bidang

- AttributeName— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama dari atribut snapshot klaster DB manual.

Atribut bernama `restore` mengacu pada daftar akun Amazon yang memiliki izin untuk menyalin atau memulihkan snapshot klaster DB manual. Untuk informasi selengkapnya, lihat tindakan API the section called “Memodifikasi DBCluster SnapshotAttribute”.

- AttributeValues— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nilai untuk atribut snapshot klaster DB manual.

  Jika `AttributeName` bidang diatur k`erestore`, maka elemen ini mengembalikan daftar IDs akun Amazon yang berwenang untuk menyalin atau memulihkan snapshot cluster DB manual. Jika nilai dari `all` ada dalam daftar, maka snapshot klaster DB manual bersifat publik dan tersedia untuk akun Amazon mana pun untuk disalin atau dipulihkan.

DBClusterSnapshotAttributesResult (struktur)

Berisi hasil dari panggilan sukses ke tindakan API the section called “Jelaskan DBCluster SnapshotAttributes”.

Atribut snapshot klaster DB manual digunakan untuk mengotorisasi akun Amazon lain untuk memulihkan snapshot klaster DB manual. Untuk informasi selengkapnya, lihat tindakan API the section called “Memodifikasi DBCluster SnapshotAttribute”.

Bidang

- DBClusterSnapshotAttributes— Ini adalah Array DBClusterSnapshotAttribute objek.

  Daftar atribut dan nilai untuk snapshot klaster DB manual.

- DBClusterSnapshotIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi dari snapshot klaster DB manual tempat atribut diterapkan.

`DBClusterSnapshotAttributesResult` digunakan sebagai elemen respons untuk:

- Jelaskan DBCluster SnapshotAttributes
- Memodifikasi DBCluster SnapshotAttribute

API Peristiwa Neptune

Tindakan:

- CreateEventSubscription (tindakan)
- DeleteEventSubscription (tindakan)
- ModifyEventSubscription (tindakan)
- DescribeEventSubscriptions (tindakan)
- AddSourceIdentifierToSubscription (tindakan)
- RemoveSourceIdentifierFromSubscription (tindakan)
- DescribeEvents (tindakan)
- DescribeEventCategories (tindakan)

Struktur:

- Peristiwa (struktur)
- EventCategoriesMap (struktur)
- EventSubscription (struktur)

CreateEventSubscription (tindakan)

> Nama AWS CLI untuk API ini adalah:. `create-event-subscription`

Membuat langganan notifikasi peristiwa. Tindakan ini memerlukan topik Amazon Resource Name (ARN) yang dibuat dengan menggunakan konsol Neptune, konsol SNS, atau API SNS. Untuk mendapatkan ARN dengan SNS, Anda harus membuat topik di Amazon SNS dan berlangganan topik tersebut. ARN ditampilkan di konsol SNS.

Anda dapat menentukan jenis source (SourceType) yang ingin diberitahukan, memberikan daftar sumber Neptunus SourceIds () yang memicu peristiwa, dan memberikan daftar kategori peristiwa EventCategories () untuk peristiwa yang ingin Anda beri tahu. Misalnya, Anda dapat menentukan SourceType = db-instance, = SourceIds mydbinstance1, mydbinstance2 dan = Availability, Backup. EventCategories

Jika Anda menentukan SourceType dan SourceIds, seperti SourceType = db-instance dan SourceIdentifier = myDBInstance1, Anda akan diberi tahu tentang semua kejadian db-instance

untuk sumber yang ditentukan. Jika Anda menentukan SourceType tetapi tidak menentukan a SourceIdentifier, Anda menerima pemberitahuan peristiwa untuk jenis sumber tersebut untuk semua sumber Neptunus Anda. Jika Anda tidak menentukan baik SourceType maupun SourceIdentifier, Anda akan diberitahu tentang peristiwa yang dihasilkan dari semua sumber Neptunus milik akun pelanggan Anda.

Permintaan

- Enabled(dalam CLI:`--enabled`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai Boolean; atur ke betul untuk mengaktifkan langganan, atur ke salah untuk membuat langganan tetapi tanpa mengaktifkannya.
- EventCategories(dalam CLI:`--event-categories`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar kategori acara untuk SourceType yang ingin Anda berlangganan. Anda dapat melihat daftar kategori untuk diberikan SourceType dengan menggunakan DescribeEventCategoriestindakan.
- SnsTopicArn(dalam CLI:`--sns-topic-arn`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) dari topik SNS yang dibuat untuk notifikasi acara. ARN dibuat oleh Amazon SNS saat Anda membuat topik dan berlangganan topik tersebut.
- SourceIds(dalam CLI:`--source-ids`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar pengidentifikasi sumber acara yang acaranya dikembalikan. Jika tidak ditentukan, maka semua sumber disertakan dalam respons. Pengidentifikasi harus dimulai dengan huruf dan hanya boleh berisi huruf ASCII, angka, dan tanda hubung; tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung berturut-turut.

  Batasan:

  - Jika SourceIds disediakan, juga SourceType harus disediakan.
  - Jika jenis sumber adalah instans DB, maka `DBInstanceIdentifier` harus disediakan.
  - Jika jenis sumber adalah grup kemanan DB, `DBSecurityGroupName` harus disediakan.
  - Jika jenis sumber adalah grup parameter DB, `DBParameterGroupName` harus disediakan.
  - Jika jenis sumber adalah snapshot DB, `DBSnapshotIdentifier` harus disediakan.
- SourceType(dalam CLI:`--source-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

Jenis sumber yang menghasilkan acara. Misalnya, jika Anda ingin diberi tahu tentang peristiwa yang dihasilkan oleh sebuah instans DB, Anda akan mengatur parameter ini ke db-instance. Jika nilai ini tidak ditentukan, semua peristiwa akan dikembalikan.

Nilai yang valid: `db-instance` `db-cluster` | `db-parameter-group` | `db-security-group` | `db-snapshot` | `db-cluster-snapshot`

- SubscriptionName(dalam CLI:`--subscription-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama dari langganan.

  Kendala: Nama harus kurang dari 255 karakter.

- Tags(dalam CLI:`--tags`) — Sebuah array objek. Tag

  Tanda yang akan diterapkan ke langganan acara baru.

Respons

Berisi hasil pemanggilan yang berhasil dari tindakan the section called “DescribeEventSubscriptions”.

- CustomerAwsId— String, tipe: `string` (string yang dikodekan UTF-8).

  Akun pelanggan Amazon yang terkait dengan langganan notifikasi peristiwa.

- CustSubscriptionId— String, tipe: `string` (string yang dikodekan UTF-8).

  Id langganan notifikasi peristiwa.

- Enabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai Boolean yang menunjukkan apakah langganan diaktifkan. True menunjukkan bahwa langganan diaktifkan.

- EventCategoriesList— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar kategori peristiwa untuk langganan notifikasi peristiwa.

- EventSubscriptionArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk langganan peristiwa.

- SnsTopicArn— String, tipe: `string` (string yang dikodekan UTF-8).

ARN topik langganan notifikasi peristiwa.

- SourceIdsList— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar sumber IDs untuk langganan pemberitahuan acara.

- SourceType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis sumber untuk langganan notifikasi peristiwa.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status langganan notifikasi peristiwa.

  Batasan:

  Bisa menjadi salah satu dari berikut ini: membuat | memodifikasi | menghapus | aktif | tanpa izin | topic-not-exist

  Status “no-permission” menunjukkan bahwa Neptune tidak lagi memiliki izin untuk memposting ke topik SNS. Status "topic-not-exist" menunjukkan bahwa topik telah dihapus setelah langganan dibuat.

- SubscriptionCreationTime— String, tipe: `string` (string yang dikodekan UTF-8).

  Waktu langganan notifikasi peristiwa dibuat.

Kesalahan

- EventSubscriptionQuotaExceededFault
- SubscriptionAlreadyExistFault
- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault
- SubscriptionCategoryNotFoundFault
- SourceNotFoundFault

DeleteEventSubscription (tindakan)

Nama AWS CLI untuk API ini adalah:. `delete-event-subscription`

Menghapus langganan notifikasi peristiwa.

Permintaan

- SubscriptionName(dalam CLI:`--subscription-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama langganan notifikasi peristiwa yang ingin Anda hapus.

Respons

Berisi hasil pemanggilan yang berhasil dari tindakan the section called “DescribeEventSubscriptions”.

- CustomerAwsId— String, tipe: `string` (string yang dikodekan UTF-8).

  Akun pelanggan Amazon yang terkait dengan langganan notifikasi peristiwa.
- CustSubscriptionId— String, tipe: `string` (string yang dikodekan UTF-8).

  Id langganan notifikasi peristiwa.
- Enabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai Boolean yang menunjukkan apakah langganan diaktifkan. True menunjukkan bahwa langganan diaktifkan.
- EventCategoriesList— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar kategori peristiwa untuk langganan notifikasi peristiwa.
- EventSubscriptionArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk langganan peristiwa.
- SnsTopicArn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN topik langganan notifikasi peristiwa.
- SourceIdsList— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar sumber IDs untuk langganan pemberitahuan acara.
- SourceType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis sumber untuk langganan notifikasi peristiwa.
- Status— String, tipe: `string` (string yang dikodekan UTF-8).

Status langganan notifikasi peristiwa.

Batasan:

Bisa menjadi salah satu dari berikut ini: membuat | memodifikasi | menghapus | aktif | tanpa izin | topic-not-exist

Status “no-permission” menunjukkan bahwa Neptune tidak lagi memiliki izin untuk memposting ke topik SNS. Status "topic-not-exist" menunjukkan bahwa topik telah dihapus setelah langganan dibuat.

- SubscriptionCreationTime— String, tipe: `string` (string yang dikodekan UTF-8).

  Waktu langganan notifikasi peristiwa dibuat.

Kesalahan

- SubscriptionNotFoundFault
- InvalidEventSubscriptionStateFault

ModifyEventSubscription (tindakan)

Nama AWS CLI untuk API ini adalah:. `modify-event-subscription`

Memodifikasi langganan notifikasi peristiwa yang ada. Perhatikan bahwa Anda tidak dapat mengubah pengidentifikasi sumber menggunakan panggilan ini; untuk mengubah pengidentifikasi sumber untuk langganan, gunakan panggilan the section called “AddSourceIdentifierToSubscription” dan the section called “RemoveSourceIdentifierFromSubscription”.

Anda dapat melihat daftar kategori acara untuk diberikan SourceType dengan menggunakan DescribeEventCategoriestindakan.

Permintaan

- Enabled(dalam CLI:`--enabled`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai Boolean; atur ke true untuk mengaktifkan langganan.

- EventCategories(dalam CLI:`--event-categories`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar kategori acara untuk SourceType yang ingin Anda berlangganan. Anda dapat melihat daftar kategori untuk diberikan SourceType dengan menggunakan DescribeEventCategoriestindakan.
- SnsTopicArn(dalam CLI:`--sns-topic-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) dari topik SNS yang dibuat untuk notifikasi acara. ARN dibuat oleh Amazon SNS saat Anda membuat topik dan berlangganan topik tersebut.
- SourceType(dalam CLI:`--source-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis sumber yang menghasilkan acara. Misalnya, jika Anda ingin diberi tahu tentang peristiwa yang dihasilkan oleh sebuah instans DB, Anda akan mengatur parameter ini ke db-instance. Jika nilai ini tidak ditentukan, semua peristiwa akan dikembalikan.

  Nilai yang valid: db-instance | | db-parameter-group | db-security-group db-snapshot
- SubscriptionName(dalam CLI:`--subscription-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama langganan notifikasi peristiwa.

Respons

Berisi hasil pemanggilan yang berhasil dari tindakan the section called “DescribeEventSubscriptions”.

- CustomerAwsId— String, tipe: `string` (string yang dikodekan UTF-8).

  Akun pelanggan Amazon yang terkait dengan langganan notifikasi peristiwa.
- CustSubscriptionId— String, tipe: `string` (string yang dikodekan UTF-8).

  Id langganan notifikasi peristiwa.
- Enabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai Boolean yang menunjukkan apakah langganan diaktifkan. True menunjukkan bahwa langganan diaktifkan.
- EventCategoriesList— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar kategori peristiwa untuk langganan notifikasi peristiwa.

- EventSubscriptionArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk langganan peristiwa.

- SnsTopicArn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN topik langganan notifikasi peristiwa.

- SourceIdsList— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar sumber IDs untuk langganan pemberitahuan acara.

- SourceType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis sumber untuk langganan notifikasi peristiwa.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status langganan notifikasi peristiwa.

  Batasan:

  Bisa menjadi salah satu dari berikut ini: membuat | memodifikasi | menghapus | aktif | tanpa izin | topic-not-exist

  Status “no-permission” menunjukkan bahwa Neptune tidak lagi memiliki izin untuk memposting ke topik SNS. Status "topic-not-exist" menunjukkan bahwa topik telah dihapus setelah langganan dibuat.

- SubscriptionCreationTime— String, tipe: `string` (string yang dikodekan UTF-8).

  Waktu langganan notifikasi peristiwa dibuat.

Kesalahan

- EventSubscriptionQuotaExceededFault
- SubscriptionNotFoundFault
- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault
- SubscriptionCategoryNotFoundFault

DescribeEventSubscriptions (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-event-subscriptions`

Mencantumkan semua deskripsi langganan untuk akun pelanggan. Deskripsi untuk langganan meliputi SubscriptionName, SNSTopic ARN, CustomerID,, SourceID, SourceType, dan Status. CreationTime

Jika Anda menentukan SubscriptionName, daftar deskripsi untuk langganan itu.

Permintaan

- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Parameter ini saat ini tidak didukung.
- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan DescribeOrderable DBInstance Opsi sebelumnya. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.
- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

  Default: 100

  Kendala: Minimal 20, maksimum 100.
- SubscriptionName(dalam CLI:`--subscription-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama langganan notifikasi peristiwa yang ingin Anda jelaskan.

Respons

- EventSubscriptionsList – Susunan objek EventSubscription.

  Daftar tipe EventSubscriptions data.

- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan DescribeOrderable DBInstance Opsi sebelumnya. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

Kesalahan

- SubscriptionNotFoundFault

AddSourceIdentifierToSubscription (tindakan)

Nama AWS CLI untuk API ini adalah:. `add-source-identifier-to-subscription`

Menambahkan pengidentifikasi sumber ke langganan notifikasi peristiwa yang ada.

Permintaan

- SourceIdentifier(dalam CLI:`--source-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi sumber peristiwa yang akan ditambahkan.

  Batasan:

  - Jika jenis sumber adalah instans DB, maka `DBInstanceIdentifier` harus disediakan.
  - Jika jenis sumber adalah grup kemanan DB, `DBSecurityGroupName` harus disediakan.
  - Jika jenis sumber adalah grup parameter DB, `DBParameterGroupName` harus disediakan.
  - Jika jenis sumber adalah snapshot DB, `DBSnapshotIdentifier` harus disediakan.
- SubscriptionName(dalam CLI:`--subscription-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama langganan notifikasi peristiwa yang ingin Anda tambahkan pengidentifikasi sumbernya.

Respons

Berisi hasil pemanggilan yang berhasil dari tindakan the section called “DescribeEventSubscriptions”.

- CustomerAwsId— String, tipe: `string` (string yang dikodekan UTF-8).

Akun pelanggan Amazon yang terkait dengan langganan notifikasi peristiwa.

- CustSubscriptionId— String, tipe: `string` (string yang dikodekan UTF-8).

  Id langganan notifikasi peristiwa.

- Enabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai Boolean yang menunjukkan apakah langganan diaktifkan. True menunjukkan bahwa langganan diaktifkan.

- EventCategoriesList— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar kategori peristiwa untuk langganan notifikasi peristiwa.

- EventSubscriptionArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk langganan peristiwa.

- SnsTopicArn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN topik langganan notifikasi peristiwa.

- SourceIdsList— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar sumber IDs untuk langganan pemberitahuan acara.

- SourceType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis sumber untuk langganan notifikasi peristiwa.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status langganan notifikasi peristiwa.

  Batasan:

  Bisa menjadi salah satu dari berikut ini: membuat | memodifikasi | menghapus | aktif | tanpa izin | topic-not-exist

  Status “no-permission” menunjukkan bahwa Neptune tidak lagi memiliki izin untuk memposting ke topik SNS. Status "topic-not-exist" menunjukkan bahwa topik telah dihapus setelah langganan dibuat.

- SubscriptionCreationTime— String, tipe: `string` (string yang dikodekan UTF-8).

  Waktu langganan notifikasi peristiwa dibuat.

Kesalahan

- SubscriptionNotFoundFault
- SourceNotFoundFault

RemoveSourceIdentifierFromSubscription (tindakan)

Nama AWS CLI untuk API ini adalah:. `remove-source-identifier-from-subscription`

Menghapus pengidentifikasi sumber dari langganan notifikasi peristiwa yang ada.

Permintaan

- SourceIdentifier(dalam CLI:`--source-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi sumber yang akan dihapus dari langganan, seperti pengidentifikasi instans DB untuk instans DB, atau nama grup keamanan.
- SubscriptionName(dalam CLI:`--subscription-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama langganan notifikasi peristiwa yang ingin Anda hapus pengidentifikasi sumbernya.

Respons

Berisi hasil pemanggilan yang berhasil dari tindakan the section called “DescribeEventSubscriptions”.

- CustomerAwsId— String, tipe: `string` (string yang dikodekan UTF-8).

  Akun pelanggan Amazon yang terkait dengan langganan notifikasi peristiwa.
- CustSubscriptionId— String, tipe: `string` (string yang dikodekan UTF-8).

  Id langganan notifikasi peristiwa.
- Enabled— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai Boolean yang menunjukkan apakah langganan diaktifkan. True menunjukkan bahwa langganan diaktifkan.
- EventCategoriesList— String, tipe: `string` (string yang dikodekan UTF-8).

Daftar kategori peristiwa untuk langganan notifikasi peristiwa.

- EventSubscriptionArn— String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk langganan peristiwa.

- SnsTopicArn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN topik langganan notifikasi peristiwa.

- SourceIdsList— String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar sumber IDs untuk langganan pemberitahuan acara.

- SourceType— String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis sumber untuk langganan notifikasi peristiwa.

- Status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status langganan notifikasi peristiwa.

  Batasan:

  Bisa menjadi salah satu dari berikut ini: membuat | memodifikasi | menghapus | aktif | tanpa izin | topic-not-exist

  Status “no-permission” menunjukkan bahwa Neptune tidak lagi memiliki izin untuk memposting ke topik SNS. Status "topic-not-exist" menunjukkan bahwa topik telah dihapus setelah langganan dibuat.

- SubscriptionCreationTime— String, tipe: `string` (string yang dikodekan UTF-8).

  Waktu langganan notifikasi peristiwa dibuat.

Kesalahan

- SubscriptionNotFoundFault
- SourceNotFoundFault

DescribeEvents (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-events`

Mengembalikan peristiwa yang terkait dengan instans DB, grup keamanan DB, snapshot DB, dan grup parameter DB selama 14 hari terakhir. Peristiwa khusus untuk instans DB, grup keamanan DB, snapshot basis data, atau grup parameter DB tertentu yang dapat diperoleh dengan memberikan nama sebagai parameter. Secara default, jam terakhir peristiwa dikembalikan.

Permintaan

- Duration(dalam CLI:`--duration`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah menit untuk mengambil peristiwa.

  Default: 60

- EndTime(dalam CLI:`--end-time`) — a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Akhir interval waktu untuk mengambil peristiwa, ditentukan dalam format ISO 8601. Untuk informasi lebih lanjut tentang ISO 8601, buka halaman Wikipedia ISO86 01.

  Contoh: 2009-07-08T18:00Z

- EventCategories(dalam CLI:`--event-categories`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar kategori peristiwa yang memicu notifikasi untuk langganan notifikasi peristiwa.

- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Parameter ini saat ini tidak didukung.

- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh DescribeEvents permintaan sebelumnya. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

Default: 100

Kendala: Minimal 20, maksimum 100.

- SourceIdentifier(dalam CLI:`--source-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi sumber peristiwa yang peristiwanya dikembalikan. Jika tidak ditentukan, maka semua sumber disertakan dalam respons.

  Batasan:

  - Jika SourceIdentifier disediakan, juga SourceType harus disediakan.
  - Jika jenis sumber adalah `DBInstance`, maka `DBInstanceIdentifier` harus disediakan.
  - Jika jenis sumber adalah `DBSecurityGroup`, sebuah `DBSecurityGroupName` harus disediakan.
  - Jika jenis sumber adalah `DBParameterGroup`, sebuah `DBParameterGroupName` harus disediakan.
  - Jika jenis sumber adalah `DBSnapshot`, sebuah `DBSnapshotIdentifier` harus disediakan.
  - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
- SourceType(dalam CLI:`--source-type`) — a SourceType, dari tipe: `string` (string yang dikodekan UTF-8).

  Sumber peristiwa untuk mengambil peristiwa. Jika tidak ada nilai yang ditentukan, semua peristiwa dikembalikan.
- StartTime(dalam CLI:`--start-time`) — a TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Awal interval waktu untuk mengambil peristiwa, ditentukan dalam format ISO 8601. Untuk informasi lebih lanjut tentang ISO 8601, buka halaman Wikipedia ISO86 01.

  Contoh: 2009-07-08T18:00Z

Respons

- Events – Susunan objek Peristiwa.

  Daftar instans the section called “Peristiwa”.

- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination (pemberian nomor halaman) opsional yang disediakan oleh permintaan Peristiwa sebelumnya. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

DescribeEventCategories (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-event-categories`

Menampilkan daftar kategori untuk semua jenis sumber peristiwa, atau, jika ditentukan, untuk jenis sumber tertentu.

Permintaan

- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Parameter ini saat ini tidak didukung.
- SourceType(dalam CLI:`--source-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis sumber yang menghasilkan acara.

  Nilai yang valid: db-instance | | db-parameter-group | db-security-group db-snapshot

Respons

- EventCategoriesMapList – Susunan objek EventCategoriesMap.

  Daftar tipe EventCategoriesMap data.

Struktur:

Peristiwa (struktur)

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “DescribeEvents”.

Bidang

- Date— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

Menentukan tanggal dan waktu peristiwa.

- EventCategories— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan kategori untuk peristiwa tersebut.

- Message— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan teks peristiwa ini.

- SourceArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk peristiwa.

- SourceIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menyediakan pengidentifikasi untuk sumber peristiwa.

- SourceType— Ini adalah SourceType, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan jenis sumber untuk peristiwa ini.

EventCategoriesMap (struktur)

Berisi hasil pemanggilan yang berhasil dari tindakan the section called “DescribeEventCategories”.

Bidang

- EventCategories— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Kategori peristiwa untuk jenis sumber yang ditentukan

- SourceType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis sumber yang dimiliki oleh kategori yang dikembalikan

EventSubscription (struktur)

Berisi hasil pemanggilan yang berhasil dari tindakan the section called “DescribeEventSubscriptions”.

Bidang

- CustomerAwsId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

Akun pelanggan Amazon yang terkait dengan langganan notifikasi peristiwa.

- CustSubscriptionId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Id langganan notifikasi peristiwa.

- Enabled— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai Boolean yang menunjukkan apakah langganan diaktifkan. True menunjukkan bahwa langganan diaktifkan.

- EventCategoriesList— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar kategori peristiwa untuk langganan notifikasi peristiwa.

- EventSubscriptionArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Amazon Resource Name (ARN) untuk langganan peristiwa.

- SnsTopicArn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  ARN topik langganan notifikasi peristiwa.

- SourceIdsList— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar sumber IDs untuk langganan pemberitahuan acara.

- SourceType— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis sumber untuk langganan notifikasi peristiwa.

- Status— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Status langganan notifikasi peristiwa.

  Batasan:

  Bisa menjadi salah satu dari berikut ini: membuat | memodifikasi | menghapus | aktif | tanpa izin | topic-not-exist

  Status “no-permission” menunjukkan bahwa Neptune tidak lagi memiliki izin untuk memposting ke topik SNS. Status "topic-not-exist" menunjukkan bahwa topik telah dihapus setelah langganan dibuat.

- SubscriptionCreationTime— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Waktu langganan notifikasi peristiwa dibuat.

`EventSubscription` digunakan sebagai elemen respons untuk:

- CreateEventSubscription
- ModifyEventSubscription
- AddSourceIdentifierToSubscription
- RemoveSourceIdentifierFromSubscription
- DeleteEventSubscription

Neptunus lainnya APIs

Tindakan:

- AddTagsToResource (tindakan)
- ListTagsForResource (tindakan)
- RemoveTagsFromResource (tindakan)
- ApplyPendingMaintenanceAction (tindakan)
- DescribePendingMaintenanceActions (tindakan)
- Jelaskan DBEngine Versi (tindakan)

Struktur:

- DBEngineVersi (struktur)
- EngineDefaults (struktur)
- PendingMaintenanceAction (struktur)
- ResourcePendingMaintenanceActions (struktur)
- UpgradeTarget (struktur)
- Tag (Struktur)

AddTagsToResource (tindakan)

Nama AWS CLI untuk API ini adalah:. `add-tags-to-resource`

Menambahkan tag metadata ke sumber daya Amazon Neptune. Tag ini juga dapat digunakan dengan alokasi biaya pelaporan untuk melacak biaya yang terkait dengan sumber daya Amazon Neptune, atau digunakan dalam pernyataan Kondisi dalam kebijakan IAM untuk Amazon Neptune.

Permintaan

- ResourceName(dalam CLI:`--resource-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Sumber daya Amazon Neptune tempat tag ditambahkan. Nilai ini adalah sebuah Amazon Resource Name (ARN). Untuk informasi tentang membuat ARN, lihat Membangun Amazon Resource Name (ARN).
- Tags(dalam CLI:`--tags`) - Diperlukan: Sebuah array objek. Tag

  Tag yang akan ditetapkan ke sumber daya Amazon Neptune.

Respons

- Tidak ada parameter Respons.

Kesalahan

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

ListTagsForResource (tindakan)

Nama AWS CLI untuk API ini adalah:. `list-tags-for-resource`

Mencantumkan semua tanda pada sumber daya Amazon Neptune.

Permintaan

- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Parameter ini saat ini tidak didukung.

- ResourceName(dalam CLI:`--resource-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Sumber daya Amazon Neptune dengan tag untuk dicantumkan. Nilai ini adalah sebuah Amazon Resource Name (ARN). Untuk informasi tentang membuat ARN, lihat Membangun Amazon Resource Name (ARN).

Respons

- TagList – Susunan objek Tag.

  Daftar tag yang dikembalikan oleh ListTagsForResource operasi.

Kesalahan

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

RemoveTagsFromResource (tindakan)

Nama AWS CLI untuk API ini adalah:. `remove-tags-from-resource`

Menghapus tanda metadata dari sumber daya Amazon Neptune.

Permintaan

- ResourceName(dalam CLI:`--resource-name`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Sumber daya Amazon Neptune tempat tag dihapus. Nilai ini adalah sebuah Amazon Resource Name (ARN). Untuk informasi tentang membuat ARN, lihat Membangun Amazon Resource Name (ARN).
- TagKeys(dalam CLI:`--tag-keys`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kunci tag (nama) dari tag yang akan dihapus.

Respons

- Tidak ada parameter Respons.

Kesalahan

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

ApplyPendingMaintenanceAction (tindakan)

Nama AWS CLI untuk API ini adalah:. `apply-pending-maintenance-action`

Menerapkan tindakan pemeliharaan tertunda ke sumber daya (misalnya, ke instans DB).

Permintaan

- ApplyAction(dalam CLI:`--apply-action`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Tindakan pemeliharaan tertunda untuk diterapkan ke sumber daya ini.

  Nilai valid: `system-update`, `db-upgrade`
- OptInType(dalam CLI:`--opt-in-type`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nilai yang menentukan jenis permintaan opt-in atau membatalkan permintaan opt-in. Permintaan opt-in dari jenis `immediate` tidak dapat dibatalkan.

  Nilai valid:

  - `immediate` - Terapkan tindakan pemeliharaan segera.
  - `next-maintenance` - Terapkan tindakan pemeliharaan selama jendela pemeliharaan berikutnya untuk sumber daya.
  - `undo-opt-in` - Batalkan permintaan opt-in `next-maintenance` apa pun yang ada.
- ResourceIdentifier(dalam CLI:`--resource-identifier`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

Amazon Resource Name (ARN) dari sumber daya tempat tindakan pemeliharaan tertunda diterapkan. Untuk informasi tentang membuat ARN, lihat Membangun Amazon Resource Name (ARN).

Respons

Menjelaskan tindakan pemeliharaan tertunda untuk sumber daya.

- PendingMaintenanceActionDetails – Susunan objek PendingMaintenanceAction.

  Daftar yang menyediakan detail tentang tindakan pemeliharaan yang tertunda untuk sumber daya.
- ResourceIdentifier— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari sumber daya yang memiliki tindakan pemeliharaan tertunda.

Kesalahan

- ResourceNotFoundFault

DescribePendingMaintenanceActions (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-pending-maintenance-actions`

Mengembalikan daftar sumber daya (misalnya, instans DB) yang memiliki setidaknya satu tindakan pemeliharaan tertunda.

Permintaan

- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Filter yang menentukan satu sumber daya atau lebih untuk mengembalikan tindakan pemeliharaan tertunda.

  Filter yang didukung:

  - `db-cluster-id`- Menerima pengidentifikasi cluster DB dan DB cluster Amazon Resource Names ()ARNs. Daftar hasil hanya akan menyertakan tindakan pemeliharaan yang tertunda untuk cluster DB yang diidentifikasi oleh ini ARNs.

  - `db-instance-id`- Menerima pengidentifikasi instans DB dan instans DB. ARNs Daftar hasil hanya akan menyertakan tindakan pemeliharaan yang tertunda untuk instans DB yang diidentifikasi oleh ini ARNs.
- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan `DescribePendingMaintenanceActions` sebelumnya. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga jumlah catatn ditentukan oleh `MaxRecords`.
- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika lebih banyak catatan ada daripada nilai `MaxRecords` yang ditentukan, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil yang tersisa dapat diambil.

  Default: 100

  Kendala: Minimal 20, maksimum 100.
- ResourceIdentifier(dalam CLI:`--resource-identifier`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari sumber daya untuk mengembalikan tindakan pemeliharaan yang tertunda.

Respons

- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan `DescribePendingMaintenanceActions` sebelumnya. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga jumlah catatn ditentukan oleh `MaxRecords`.
- PendingMaintenanceActions – Susunan objek ResourcePendingMaintenanceActions.

  Daftar tindakan pemeliharaan yang tertunda untuk sumber daya.

Kesalahan

- ResourceNotFoundFault

Jelaskan DBEngine Versi (tindakan)

Nama AWS CLI untuk API ini adalah:. `describe-db-engine-versions`

Mengembalikan daftar dari mesin DB yang tersedia.

Permintaan

- DBParameterGroupFamily(dalam CLI:`--db-parameter-group-family`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama keluarga grup parameter DB tertentu yang dikembalikan detailnya.

  Batasan:

  - Jika disediakan, harus cocok dengan yang sudah ada DBParameterGroupFamily.
- DefaultOnly(dalam CLI:`--default-only`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan bahwa hanya versi default dari mesin tertentu atau mesin dan kombinasi versi utama dikembalikan.
- Engine(dalam CLI:`--engine`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Mesin basis data untuk mengembalikan.
- EngineVersion(dalam CLI:`--engine-version`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Versi mesin basis data untuk mengembalikan.

  Contoh: `5.1.49`
- Filters(dalam CLI:`--filters`) — Sebuah array objek. Filter

  Saat ini tidak didukung.
- ListSupportedCharacterSets(dalam CLI:`--list-supported-character-sets`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika parameter ini ditentukan dan mesin yang diminta mendukung parameter `CharacterSetName` untuk `CreateDBInstance`, responsnya mencakup daftar set karakter yang didukung untuk setiap versi mesin.

- ListSupportedTimezones(dalam CLI:`--list-supported-timezones`) — a BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika parameter ini ditentukan dan mesin yang diminta mendukung parameter `TimeZone` untuk `CreateDBInstance`, responsnya mencakup daftar zona waktu yang didukung untuk setiap versi mesin.

- Marker(dalam CLI:`--marker`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

- MaxRecords(dalam CLI:`--max-records`) — sebuah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum catatan yang akan dikembalikan dalam respons. Jika tersedia lebih banyak daripada nilai `MaxRecords`, token pagination (pemberian nomor halaman) yang disebut penanda disertakan dalam respons sehingga hasil berikut dapat diambil.

  Default: 100

  Kendala: Minimal 20, maksimum 100.

Respons

- DBEngineVersions – Susunan objek DBEngineVersi.

  Daftar Elemen `DBEngineVersion`.

- Marker— String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh permintaan sebelumnya. Jika parameter ini ditentukan, respon hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

Struktur:

DBEngineVersi (struktur)

Tipe data ini digunakan sebagai elemen respons dalam tindakan the section called “Jelaskan DBEngine Versi”.

Bidang

- DBEngineDescription— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Deskripsi mesin basis data.
- DBEngineVersionDescription— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Deskripsi versi mesin basis data.
- DBParameterGroupFamily— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama famili grup parameter DB untuk mesin basis data.
- Engine— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama mesin basis data.
- EngineVersion— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nomor versi mesin basis data.
- ExportableLogTypes— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis log yang dimiliki mesin database untuk diekspor ke CloudWatch Log.
- SupportedTimezones— Ini adalah Array Zona waktu objek.

  Daftar zona waktu didukung oleh mesin ini untuk parameter `Timezone` dari tindakan`CreateDBInstance`.
- SupportsGlobalDatabases— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang menunjukkan apakah Anda dapat menggunakan database global Aurora dengan versi mesin DB tertentu.
- SupportsLogExportsToCloudwatchLogs— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

Nilai yang menunjukkan apakah versi engine mendukung ekspor jenis log yang ditentukan oleh ExportableLogTypes ke CloudWatch Log.

- SupportsReadReplica— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah versi mesin basis data mendukung replika baca.

- ValidUpgradeTarget— Ini adalah Array UpgradeTarget objek.

  Daftar versi mesin yang dapat ditingkatkan untuk versi mesin basis data ini.

EngineDefaults (struktur)

Berisi hasil pemanggilan yang berhasil dari tindakan the section called "DescribeEngineDefaultParameters".

Bidang

- DBParameterGroupFamily— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan nama keluarga grup parameter DB yang diterapkan parameter default mesin.

- Marker— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Token pagination opsional yang disediakan oleh EngineDefaults permintaan sebelumnya. Jika parameter ini ditentukan, respons hanya menyertakan catatan di luar penanda, hingga nilai yang ditentukan dengan `MaxRecords`.

- Parameters— Ini adalah Array Parameter objek.

  Berisi daftar parameter default mesin.

`EngineDefaults` digunakan sebagai elemen respons untuk:

- DescribeEngineDefaultParameters
- DescribeEngineDefaultClusterParameters

PendingMaintenanceAction (struktur)

Menyediakan informasi tentang tindakan pemeliharaan yang tertunda untuk sumber daya.

Bidang

- Action— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis tindakan pemeliharaan tertunda yang tersedia untuk sumber daya.

- AutoAppliedAfterDate— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tanggal jendela pemeliharaan saat tindakan diterapkan. Tindakan pemeliharaan diterapkan ke sumber daya selama jendela pemeliharaan pertama setelah tanggal ini. Jika tanggal ini ditentukan, semua permintaan opt-in `next-maintenance` diabaikan.

- CurrentApplyDate— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tanggal efektif ketika tindakan pemeliharaan tertunda diterapkan ke sumber daya. Tanggal ini mempertimbangkan permintaan opt-in yang diterima dari API the section called "ApplyPendingMaintenanceAction",`AutoAppliedAfterDate`, dan `ForcedApplyDate`. Nilai ini kosong jika permintaan opt-in belum diterima dan tidak ada yang ditetapkan sebagai `AutoAppliedAfterDate` atau `ForcedApplyDate`.

- Description— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Deskripsi yang memberikan detail lebih lanjut tentang tindakan pemeliharaan.

- ForcedApplyDate— Ini adalah TStamp, tipe: `timestamp` (titik waktu, umumnya didefinisikan sebagai offset dari tengah malam 1970-01-01).

  Tanggal ketika tindakan pemeliharaan diterapkan secara otomatis. Tindakan pemeliharaan diterapkan ke sumber daya pada tanggal ini terlepas dari jendela pemeliharaan untuk sumber daya. Jika tanggal ini ditentukan, semua permintaan opt-in `immediate` diabaikan.

- OptInStatus— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menunjukkan jenis permintaan opt-in yang telah diterima untuk sumber daya.

ResourcePendingMaintenanceActions (struktur)

Menjelaskan tindakan pemeliharaan tertunda untuk sumber daya.

Bidang

- PendingMaintenanceActionDetails— Ini adalah Array PendingMaintenanceAction objek.

  Daftar yang menyediakan detail tentang tindakan pemeliharaan yang tertunda untuk sumber daya.
- ResourceIdentifier— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari sumber daya yang memiliki tindakan pemeliharaan tertunda.

`ResourcePendingMaintenanceActions` digunakan sebagai elemen respons untuk:

- ApplyPendingMaintenanceAction

UpgradeTarget (struktur)

Versi mesin basis data yang dapat ditingkatkan instans DB.

Bidang

- AutoUpgrade— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang menunjukkan apakah versi target diterapkan ke instance DB sumber apa pun yang telah AutoMinorVersionUpgrade disetel ke true.
- Description— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Versi mesin basis data yang dapat ditingkatkan instans DB.
- Engine— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama mesin basis data target peningkatan.
- EngineVersion— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nomor versi mesin basis data target peningkatan.
- IsMajorVersionUpgrade— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Nilai yang menunjukkan apakah mesin basis data ditingkatkan ke versi major.
- SupportsGlobalDatabases— Ini adalah BooleanOptional, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

Nilai yang menunjukkan apakah Anda dapat menggunakan database global Neptunus dengan versi mesin target.

Tag (Struktur)

Metadata yang ditetapkan ke sumber daya Amazon Neptune yang terdiri dari pasangan nilai kunci.

Bidang

- Key— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Kunci adalah nama tag yang wajib diisi. Nilai string dapat mulai dari 1 hingga 128 karakter Unicode dan tidak boleh diawali dengan `aws:` atau `rds:`. String tersebut hanya dapat berisi rangkaian huruf Unicode, angka, spasi, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$").
- Value— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nilai adalah nilai tag opsional. Nilai string dapat mulai dari 1 hingga 256 karakter Unicode dan tidak boleh diawali dengan `aws:` atau `rds:`. String tersebut hanya dapat berisi rangkaian huruf Unicode, angka, spasi, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$").

Datatype Neptune Umum

Struktur:

- AvailabilityZone (struktur)
- DBSecurityGroupMembership (struktur)
- DomainMembership (struktur)
- DoubleRange (struktur)
- Titik akhir (struktur)
- Filter (struktur)
- Kisaran (struktur)
- ServerlessV2 (struktur) ScalingConfiguration
- ServerlessV2 (struktur) ScalingConfigurationInfo
- Zona waktu (struktur)
- VpcSecurityGroupMembership (struktur)

AvailabilityZone (struktur)

Menentukan Availability Zone.

Bidang

- Name— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Availability Zone.

DBSecurityGroupMembership (struktur)

Menentukan keanggotaan dalam kelompok keamanan DB yang ditunjuk.

Bidang

- DBSecurityGroupName— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama dari grup keamanan DB.
- Status— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Status grup keamanan DB.

DomainMembership (struktur)

Catatan keanggotaan Domain Direktori Active yang terkait dengan instans DB.

Bidang

- Domain— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Pengenal Domain Direktori Aktif.
- FQDN— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama domain yang sepenuhnya memenuhi syarat dari Domain Direktori Aktif.
- IAMRoleName— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama IAM role yang akan digunakan saat membuat panggilan API ke Directory Service.
- Status— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

Status keanggotaan Domain Direktori Aktif Instans DB, seperti bergabung, bergabung tertunda, gagal dll).

DoubleRange (struktur)

Kisaran nilai ganda.

Bidang

- From— Ini adalah Double, dari tipe: `double` (nomor floating-point IEEE 754 presisi ganda).

  Nilai minimum dalam kisaran.
- To— Ini adalah Double, dari tipe: `double` (nomor floating-point IEEE 754 presisi ganda).

  Nilai maksimum dalam kisaran.

Titik akhir (struktur)

Menentukan titik akhir koneksi.

Untuk struktur data yang mewakili titik akhir klaster DB Amazon Neptune, lihat `DBClusterEndpoint`.

Bidang

- Address— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan alamat DNS dari instans DB.
- HostedZoneId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Menentukan ID yang ditetapkan Amazon Route 53 saat Anda membuat zona yang di-hosting.
- Port— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menentukan port yang didengarkan oleh mesin basis data.

Filter (struktur)

Parameter ini saat ini tidak didukung.

Bidang

- Name— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Parameter ini saat ini tidak didukung.
- Values— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Parameter ini saat ini tidak didukung.

Kisaran (struktur)

Kisaran nilai integer.

Bidang

- From— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Nilai minimum dalam kisaran.
- Step— Ini adalah IntegerOptional, dari tipe: `integer` (integer 32-bit yang ditandatangani).

  Nilai langkah untuk kisaran. Misalnya, jika Anda memiliki kisaran 5.000 sampai 10.000, dengan nilai langkah 1.000, nilai yang valid mulai dari 5.000 dan naik 1.000. Meskipun 7.500 berada dalam kisaran, itu bukan nilai yang valid untuk kisaran. Nilai yang valid adalah 5.000, 6.000, 7.000, 8.000...
- To— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Nilai maksimum dalam kisaran.

ServerlessV2 (struktur) ScalingConfiguration

Berisi konfigurasi penskalaan cluster DB Neptunus Tanpa Server.

Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

Bidang

- MaxCapacity— Ini adalah DoubleOptional, dari tipe: `double` (nomor floating-point IEEE 754 presisi ganda).

Jumlah maksimum unit kapasitas Neptunus NCUs () untuk instans DB di cluster Neptunus Tanpa Server. Anda dapat menentukan nilai NCU dalam kenaikan setengah langkah, seperti 40, 40,5, 41, dan seterusnya.

- MinCapacity— Ini adalah DoubleOptional, dari tipe: `double` (nomor floating-point IEEE 754 presisi ganda).

  Jumlah minimum unit kapasitas Neptunus NCUs () untuk instans DB di cluster Neptunus Tanpa Server. Anda dapat menentukan nilai NCU dalam kenaikan setengah langkah, seperti 8, 8.5, 9, dan seterusnya.

ServerlessV2 (struktur) ScalingConfigurationInfo

Menampilkan konfigurasi penskalaan untuk cluster DB Neptunus Tanpa Server.

Untuk informasi selengkapnya, lihat Menggunakan Amazon Neptunus Tanpa Server di Panduan Pengguna Amazon Neptunus.

Bidang

- MaxCapacity— Ini adalah DoubleOptional, dari tipe: `double` (nomor floating-point IEEE 754 presisi ganda).

  Jumlah maksimum unit kapasitas Neptunus NCUs () untuk instans DB di cluster Neptunus Tanpa Server. Anda dapat menentukan nilai NCU dalam kenaikan setengah langkah, seperti 40, 40,5, 41, dan seterusnya.

- MinCapacity— Ini adalah DoubleOptional, dari tipe: `double` (nomor floating-point IEEE 754 presisi ganda).

  Jumlah minimum unit kapasitas Neptunus NCUs () untuk instans DB di cluster Neptunus Tanpa Server. Anda dapat menentukan nilai NCU dalam kenaikan setengah langkah, seperti 8, 8.5, 9, dan seterusnya.

Zona waktu (struktur)

Zona waktu yang terkait dengan the section called “DBInstance”.

Bidang

- TimezoneName— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama zona waktu.

VpcSecurityGroupMembership (struktur)

Tipe data ini digunakan sebagai elemen respons untuk kueri tentang keanggotaan grup keamanan VPC.

Bidang

- Status— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Status grup keamanan VPC.
- VpcSecurityGroupId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama grup keamanan VPC.

Pengecualian Neptunus Khusus untuk Individu APIs

Pengecualian:

- AuthorizationAlreadyExistsFault (struktur)
- AuthorizationNotFoundFault (struktur)
- AuthorizationQuotaExceededFault (struktur)
- CertificateNotFoundFault (struktur)
- DBClusterAlreadyExistsFault (struktur)
- DBClusterNotFoundFault (struktur)
- DBClusterParameterGroupNotFoundFault (struktur)
- DBClusterQuotaExceededFault (struktur)
- DBClusterRoleAlreadyExistsFault (struktur)
- DBClusterRoleNotFoundFault (struktur)
- DBClusterRoleQuotaExceededFault (struktur)

- DBClusterSnapshotAlreadyExistsFault (struktur)
- DBClusterSnapshotNotFoundFault (struktur)
- DBInstanceAlreadyExistsFault (struktur)
- DBInstanceNotFoundFault (struktur)
- DBLogFileNotFoundFault (struktur)
- DBParameterGroupAlreadyExistsFault (struktur)
- DBParameterGroupNotFoundFault (struktur)
- DBParameterGroupQuotaExceededFault (struktur)
- DBSecurityGroupAlreadyExistsFault (struktur)
- DBSecurityGroupNotFoundFault (struktur)
- DBSecurityGroupNotSupportedFault (struktur)
- DBSecurityGroupQuotaExceededFault (struktur)
- DBSnapshotAlreadyExistsFault (struktur)
- DBSnapshotNotFoundFault (struktur)
- DBSubnetGroupAlreadyExistsFault (struktur)
- DBSubnetGroupDoesNotCoverEnoughAZs (struktur)
- DBSubnetGroupNotAllowedFault (struktur)
- DBSubnetGroupNotFoundFault (struktur)
- DBSubnetGroupQuotaExceededFault (struktur)
- DBSubnetQuotaExceededFault (struktur)
- DBUpgradeDependencyFailureFault (struktur)
- DomainNotFoundFault (struktur)
- EventSubscriptionQuotaExceededFault (struktur)
- GlobalClusterAlreadyExistsFault (struktur)
- GlobalClusterNotFoundFault (struktur)
- GlobalClusterQuotaExceededFault (struktur)
- InstanceQuotaExceededFault (struktur)
- Tidak cukup DBCluster CapacityFault (struktur)
- Tidak cukup DBInstance CapacityFault (struktur)

- InsufficientStorageClusterCapacityFault (struktur)
- Tidak valid DBCluster EndpointStateFault (struktur)
- Tidak valid DBCluster SnapshotStateFault (struktur)
- Tidak valid DBCluster StateFault (struktur)
- Tidak valid DBInstance StateFault (struktur)
- Tidak valid DBParameter GroupStateFault (struktur)
- Tidak valid DBSecurity GroupStateFault (struktur)
- Tidak valid DBSnapshot StateFault (struktur)
- Tidak valid DBSubnet GroupFault (struktur)
- Tidak valid DBSubnet GroupStateFault (struktur)
- Tidak valid DBSubnet StateFault (struktur)
- InvalidEventSubscriptionStateFault (struktur)
- InvalidGlobalClusterStateFault (struktur)
- InvalidOptionGroupStateFault (struktur)
- InvalidRestoreFault (struktur)
- InvalidSubnet (struktur)
- Tidak valid VPCNetwork StateFault (struktur)
- KMSKeyNotAccessibleFault (struktur)
- OptionGroupNotFoundFault (struktur)
- PointInTimeRestoreNotEnabledFault (struktur)
- ProvisionedIopsNotAvailableInAZFault (struktur)
- ResourceNotFoundFault (struktur)
- SNSInvalidTopicFault (struktur)
- SNSNoAuthorizationFault (struktur)
- SNSTopicArnNotFoundFault (struktur)
- SharedSnapshotQuotaExceededFault (struktur)
- SnapshotQuotaExceededFault (struktur)
- SourceNotFoundFault (struktur)
- StorageQuotaExceededFault (struktur)

- StorageTypeNotSupportedFault (struktur)
- SubnetAlreadyInUse (struktur)
- SubscriptionAlreadyExistFault (struktur)
- SubscriptionCategoryNotFoundFault (struktur)
- SubscriptionNotFoundFault (struktur)

AuthorizationAlreadyExistsFault (struktur)

Kode status HTTP yang dikembalikan: 400.

CIDRIP atau grup EC2 keamanan yang ditentukan sudah diotorisasi untuk grup keamanan DB yang ditentukan.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

AuthorizationNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

CIDRIP atau grup EC2 keamanan tertentu tidak diizinkan untuk grup keamanan DB yang ditentukan.

Neptune mungkin juga tidak diotorisasi melalui IAM untuk melakukan tindakan yang diperlukan atas nama Anda.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

AuthorizationQuotaExceededFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Kuota otorisasi grup keamanan DB telah tercapai.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

CertificateNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

CertificateIdentifiertidak mengacu pada sertifikat yang ada.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBClusterAlreadyExistsFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Pengguna sudah memiliki klaster DB dengan pengidentifikasi yang diberikan.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBClusterNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

DBClusterIdentifier tidak mengacu pada cluster DB yang ada.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

Sebuah pesan yang menjelaskan detail masalah.

DBClusterParameterGroupNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

DBClusterParameterGroupNametidak mengacu pada grup parameter DB Cluster yang ada.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBClusterQuotaExceededFault (struktur)

Kode status HTTP yang dikembalikan: 403.

Pengguna berusaha untuk membuat klaster DB baru dan pengguna telah mencapai kuota klaster DB maksimum yang diperbolehkan.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBClusterRoleAlreadyExistsFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Amazon Resource Name (ARN) dari IAM role yang ditentukan sudah dikaitkan dengan klaster DB yang ditentukan.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBClusterRoleNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

Amazon Resource Name (ARN) dari IAM role yang ditentukan tidak dikaitkan dengan klaster DB yang ditentukan.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBClusterRoleQuotaExceededFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Anda telah melebihi jumlah maksimum IAM role yang dapat dikaitkan dengan klaster DB tertentu.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBClusterSnapshotAlreadyExistsFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Pengguna sudah memiliki snapshot klaster DB dengan pengidentifikasi yang diberikan.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBClusterSnapshotNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

DBClusterSnapshotIdentifiertidak mengacu pada snapshot cluster DB yang ada.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBInstanceAlreadyExistsFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Pengguna sudah memiliki instans DB dengan pengidentifikasi yang diberikan.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBInstanceNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

DBInstanceIdentifier tidak mengacu pada instans DB yang ada.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBLogFileNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

LogFileNametidak mengacu pada file log DB yang ada.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

Sebuah pesan yang menjelaskan detail masalah.

DBParameterGroupAlreadyExistsFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Grup parameter DB dengan nama yang sama sudah ada.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBParameterGroupNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

DBParameterGroupNametidak mengacu pada grup parameter DB yang ada.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBParameterGroupQuotaExceededFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Permintaan akan mengakibatkan pengguna melebihi jumlah grup parameter DB yang diperbolehkan.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBSecurityGroupAlreadyExistsFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Grup keamanan DB dengan nama yang ditentukan DBSecurityGroupNamesudah ada.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBSecurityGroupNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

DBSecurityGroupNametidak mengacu pada grup keamanan DB yang ada.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBSecurityGroupNotSupportedFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Grup keamanan DB tidak diperbolehkan untuk tindakan ini.

Bidang

- message— Ini adalah ExceptionMessage, dari tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBSecurityGroupQuotaExceededFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Permintaan akan mengakibatkan pengguna melebihi jumlah grup keamanan DB yang diperbolehkan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBSnapshotAlreadyExistsFault (struktur)

Kode status HTTP yang dikembalikan: 400.

DBSnapshotIdentifier sudah digunakan oleh snapshot yang ada.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBSnapshotNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

DBSnapshotIdentifier tidak mengacu pada snapshot DB yang ada.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBSubnetGroupAlreadyExistsFault (struktur)

Kode status HTTP yang dikembalikan: 400.

DBSubnetGroupNamesudah digunakan oleh grup subnet DB yang ada.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

Sebuah pesan yang menjelaskan detail masalah.

DBSubnetGroupDoesNotCoverEnoughAZs (struktur)

Kode status HTTP yang dikembalikan: 400.

Subnet dalam grup subnet DB harus mencakup setidaknya dua Availability Zone kecuali hanya ada satu Availability Zone.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBSubnetGroupNotAllowedFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Menunjukkan bahwa DBSubnet Grup tidak boleh ditentukan saat membuat replika baca yang terletak di wilayah yang sama dengan instance sumber.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBSubnetGroupNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

DBSubnetGroupNametidak mengacu pada grup subnet DB yang ada.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBSubnetGroupQuotaExceededFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Permintaan akan mengakibatkan pengguna melebihi jumlah grup subnet DB yang diperbolehkan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBSubnetQuotaExceededFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Permintaan akan mengakibatkan pengguna melebihi jumlah subnet yang diperbolehkan dalam grup subnet DB.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DBUpgradeDependencyFailureFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Peningkatan DB gagal karena sumber daya yang tempat DB bergantung tidak dapat diubah.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

DomainNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

Domain tidak merujuk ke Domain Direktori Aktif yang ada.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

EventSubscriptionQuotaExceededFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Anda telah melampaui jumlah langganan peristiwa acara.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

GlobalClusterAlreadyExistsFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Parameter `GlobalClusterIdentifier` sudah ada. Pilih pengidentifikasi database global baru (nama unik) untuk membuat cluster database global baru.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

GlobalClusterNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

`GlobalClusterIdentifier`Itu tidak mengacu pada cluster database global yang ada.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

GlobalClusterQuotaExceededFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Jumlah cluster database global untuk akun ini sudah maksimal yang diizinkan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

InstanceQuotaExceededFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Permintaan akan mengakibatkan pengguna melebihi jumlah instans DB yang diperbolehkan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

Tidak cukup DBCluster CapacityFault (struktur)

Kode status HTTP yang dikembalikan: 403.

Klaster DB tidak memiliki kapasitas yang cukup untuk operasi saat ini.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

Tidak cukup DBInstance CapacityFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Instans DB yang ditentukan tidak tersedia di Availability Zone yang ditentukan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

InsufficientStorageClusterCapacityFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Penyimpanan yang tersedia tidak cukup untuk tindakan saat ini. Anda mungkin dapat mengatasi kesalahan ini dengan memperbarui grup subnet Anda untuk menggunakan Availability Zone berbeda yang memiliki lebih banyak penyimpanan yang tersedia.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

Tidak valid DBCluster EndpointStateFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Operasi yang diminta tidak dapat dilakukan pada titik akhir saat titik akhir dalam keadaan ini.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

Tidak valid DBCluster SnapshotStateFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Nilai yang disediakan bukan status snapshot Klaster DB yang valid.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

Tidak valid DBCluster StateFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Klaster DB tidak dalam status valid.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

Tidak valid DBInstance StateFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Instans DB yang ditentukan tidak dalam status tersedia.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

Tidak valid DBParameter GroupStateFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Grup parameter DB sedang digunakan atau dalam status tidak valid. Jika Anda mencoba menghapus grup parameter, Anda tidak dapat menghapusnya ketika grup parameter berada dalam keadaan ini.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

Tidak valid DBSecurity GroupStateFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Status grup keamanan DB tidak mengizinkan penghapusan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

Tidak valid DBSnapshot StateFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Status snapshot DB tidak mengizinkan penghapusan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

Tidak valid DBSubnet GroupFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Menunjukkan DBSubnet Grup tidak termasuk dalam VPC yang sama dengan replika baca lintas wilayah yang ada dari contoh sumber yang sama.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

Sebuah pesan yang menjelaskan detail masalah.

Tidak valid DBSubnet GroupStateFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Grup subnet DB tidak dapat dihapus karena sedang digunakan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

Tidak valid DBSubnet StateFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Subnet DB tidak dalam status tersedia.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

InvalidEventSubscriptionStateFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Langganan peristiwa dalam status tidak valid.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

InvalidGlobalClusterStateFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Kluster global berada dalam keadaan tidak valid dan tidak dapat melakukan operasi yang diminta.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

InvalidOptionGroupStateFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Grup opsi tidak dalam status tersedia.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

InvalidRestoreFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Tidak dapat memulihkan dari cadangan vpc untuk instans DB non-vpc.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

InvalidSubnet (struktur)

Kode status HTTP yang dikembalikan: 400.

Subnet yang diminta tidak valid, atau beberapa subnet yang diminta tidak semuanya berada dalam virtual private cloud (VPC) yang umum.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

Tidak valid VPCNetwork StateFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Grup subnet DB tidak mencakup semua Availability Zone setelah dibuat karena perubahan yang dibuat pengguna.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

KMSKeyNotAccessibleFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Kesalahan saat mengakses kunci KMS.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

OptionGroupNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

Grup opsi yang ditunjuk tidak dapat ditemukan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

PointInTimeRestoreNotEnabledFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Source DBInstance Identifier mengacu pada instance DB dengan BackupRetentionPeriodsama dengan 0.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

ProvisionedIopsNotAvailableInAZFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Provisioned IOPS tidak tersedia di Availability Zone yang ditentukan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

ResourceNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

ID sumber daya yang ditentukan tidak ditemukan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

Sebuah pesan yang menjelaskan detail masalah.

SNSInvalidTopicFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Topik SNS tidak valid.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

SNSNoAuthorizationFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Tidak ada otorisasi SNS.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

SNSTopicArnNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

ARN topik SNS tidak dapat ditemukan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

SharedSnapshotQuotaExceededFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Anda telah melampaui jumlah maksimum akun yang dapat Anda bagikan snapshot DB manual.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

SnapshotQuotaExceededFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Permintaan akan mengakibatkan pengguna melebihi jumlah snapshot DB yang diperbolehkan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

SourceNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

Sumber tidak dapat ditemukan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

StorageQuotaExceededFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Permintaan akan mengakibatkan pengguna melebihi jumlah penyimpanan yang tersedia yang diperbolehkan di semua instans DB.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

StorageTypeNotSupportedFault (struktur)

Kode status HTTP yang dikembalikan: 400.

StorageTypeditentukan tidak dapat dikaitkan dengan Instans DB.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

SubnetAlreadyInUse (struktur)

Kode status HTTP yang dikembalikan: 400.

Subnet DB sudah digunakan di Availability Zone.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

SubscriptionAlreadyExistFault (struktur)

Kode status HTTP yang dikembalikan: 400.

Langganan ini sudah ada.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

SubscriptionCategoryNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

Kategori langganan yang ditunjuk tidak dapat ditemukan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

SubscriptionNotFoundFault (struktur)

Kode status HTTP yang dikembalikan: 404.

Langganan yang ditunjuk tidak dapat ditemukan.

Bidang

- message— Ini adalah ExceptionMessage, tipe: `string` (string yang dikodekan UTF-8).

  Sebuah pesan yang menjelaskan detail masalah.

Referensi API Data Amazon Neptunus

Bab ini mendokumentasikan operasi API data Neptunus yang dapat Anda gunakan untuk memuat, mengakses, dan memelihara data dalam grafik Neptunus Anda.

API data Neptunus menyediakan dukungan SDK untuk memuat data, menjalankan kueri, mendapatkan informasi tentang data Anda, dan menjalankan operasi pembelajaran mesin. Ini mendukung bahasa query Gremlin dan OpenCypher di Neptunus, dan tersedia dalam semua bahasa SDK. Ini secara otomatis menandatangani permintaan API dan sangat menyederhanakan integrasi Neptunus ke dalam aplikasi.

Daftar Isi

- Mesin dataplane Neptunus, reset cepat, dan struktur umum APIs
  - GetEngineStatus (tindakan)
  - ExecuteFastReset (tindakan)
  - Struktur operasi mesin:
  - QueryLanguageVersion (struktur)
  - FastResetToken (struktur)
- Kueri Neptunus APIs
  - ExecuteGremlinQuery (tindakan)
  - ExecuteGremlinExplainQuery (tindakan)
  - ExecuteGremlinProfileQuery (tindakan)
  - ListGremlinQueries (tindakan)
  - GetGremlinQueryStatus (tindakan)
  - CancelGremlinQuery (tindakan)
  - Tindakan kueri OpenCypher:
  - ExecuteOpenCypherQuery (tindakan)
  - ExecuteOpenCypherExplainQuery (tindakan)
  - ListOpenCypherQueries (tindakan)
  - GetOpenCypherQueryStatus (tindakan)
  - CancelOpenCypherQuery (tindakan)
  - Struktur kueri:

- QueryEvalStats (struktur)
- GremlinQueryStatus (struktur)
- GremlinQueryStatusAttributes (struktur)

- Pemuat massal pesawat data Neptunus APIs
  - StartLoaderJob (tindakan)
  - GetLoaderJobStatus (tindakan)
  - ListLoaderJobs (tindakan)
  - CancelLoaderJob (tindakan)
  - Struktur beban massal:
  - LoaderIdResult (struktur)
- Neptunus mengalirkan API jalur data
  - GetPropertygraphStream (tindakan)
  - Struktur data aliran:
  - PropertygraphRecord (struktur)
  - PropertygraphData (struktur)
- Statistik jalur data Neptunus dan ringkasan grafik APIs
  - GetPropertygraphStatistics (tindakan)
  - ManagePropertygraphStatistics (tindakan)
  - DeletePropertygraphStatistics (tindakan)
  - GetPropertygraphSummary (tindakan)
  - Struktur statistik:
  - Statistik (struktur)
  - StatisticsSummary (struktur)
  - DeleteStatisticsValueMap (struktur)
  - RefreshStatisticsIdMap (struktur)
  - NodeStructure (struktur)
  - EdgeStructure (struktur)
  - SubjectStructure (struktur)
  - PropertygraphSummaryValueMap (struktur)
  - PropertygraphSummary (struktur)

- API pemrosesan data Neptunus Neptunus
  - Mulai MLData ProcessingJob (tindakan)
  - Daftar MLData ProcessingJobs (tindakan)
  - Dapatkan MLData ProcessingJob (tindakan)
  - Batalkan MLData ProcessingJob (tindakan)
  - Struktur tujuan umum ML:
  - MlResourceDefinition (struktur)
  - MlConfigDefinition (struktur)
- API pelatihan model Neptunus Neptunus
  - Mulai MLModel TrainingJob (tindakan)
  - Daftar MLModel TrainingJobs (tindakan)
  - Dapatkan MLModel TrainingJob (tindakan)
  - Batalkan MLModel TrainingJob (tindakan)
  - Struktur pelatihan model:
  - CustomModelTrainingParameters (struktur)
- Model Neptunus Neptunus mengubah API
  - Mulai MLModel TransformJob (tindakan)
  - Daftar MLModel TransformJobs (tindakan)
  - Dapatkan MLModel TransformJob (tindakan)
  - Batalkan MLModel TransformJob (tindakan)
  - Struktur transformasi model:
  - CustomModelTransformParameters (struktur)
- API titik akhir inferensi Neptunus ML
  - Buat MLEndpoint (tindakan)
  - Daftar MLEndpoints (tindakan)
  - Dapatkan MLEndpoint (tindakan)
  - Hapus MLEndpoint (tindakan)
- Pengecualian API jalur data Neptunus
  - AccessDeniedException (struktur)
  - BadRequestException (struktur)

- BulkLoadIdNotFoundException (struktur)
- CancelledByUserException (struktur)
- ClientTimeoutException (struktur)
- ConcurrentModificationException (struktur)
- ConstraintViolationException (struktur)
- ExpiredStreamException (struktur)
- FailureByQueryException (struktur)
- IllegalArgumentException (struktur)
- InternalFailureException (struktur)
- InvalidArgumentException (struktur)
- InvalidNumericDataException (struktur)
- InvalidParameterException (struktur)
- LoadUrlAccessDeniedException (struktur)
- MalformedQueryException (struktur)
- MemoryLimitExceededException (struktur)
- MethodNotAllowedException (struktur)
- MissingParameterException (struktur)
- MLResourceNotFoundException (struktur)
- ParsingException (struktur)
- PreconditionsFailedException (struktur)
- QueryLimitExceededException (struktur)
- QueryLimitException (struktur)
- QueryTooLargeException (struktur)
- ReadOnlyViolationException (struktur)
- S3Exception (struktur)
- ServerShutdownException (struktur)
- StatisticsNotAvailableException (struktur)
- StreamRecordsNotFoundException (struktur)
- ThrottlingException (struktur)
- TimeLimitExceededException (struktur)

- TooManyRequestsException (struktur)
- UnsupportedOperationException (struktur)
- UnloadUrlAccessDeniedException (struktur)

Mesin dataplane Neptunus, reset cepat, dan struktur umum APIs

Operasi mesin:

- GetEngineStatus (tindakan)
- ExecuteFastReset (tindakan)

Struktur operasi mesin:

- QueryLanguageVersion (struktur)
- FastResetToken (struktur)

GetEngineStatus (tindakan)

Nama AWS CLI untuk API ini adalah:. `get-engine-status`

Mengambil status database grafik pada host.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan otentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. GetEngineStatus

Permintaan

- Tidak ada parameter Permintaan.

Respons

- dbEngineVersion— String, tipe: `string` (string yang dikodekan UTF-8).

  Setel ke versi mesin Neptunus yang berjalan di cluster DB Anda. Jika versi mesin ini telah di-patch secara manual sejak dirilis, nomor versi diawali dengan `Patch-`.
- dfeQueryEngine— String, tipe: `string` (string yang dikodekan UTF-8).

Setel ke `enabled` jika mesin DFE sepenuhnya diaktifkan, atau ke `viaQueryHint` (default) jika mesin DFE hanya digunakan dengan kueri yang memiliki petunjuk kueri yang disetel keuseDFE. `true`

- features— Ini adalah array peta pasangan kunci-nilai di mana:

  Setiap kunci adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Setiap nilai adalah Dokumen, tipe: `document` (konten terbuka protokol-agnostik yang diwakili oleh model data seperti JSON).

  Berisi informasi status tentang fitur yang diaktifkan pada cluster DB Anda.
- gremlin — Sebuah objek QueryLanguageVersion.

  Berisi informasi tentang bahasa kueri Gremlin yang tersedia di cluster Anda. Secara khusus, ini berisi bidang versi yang menentukan TinkerPop versi saat ini yang digunakan oleh mesin.
- labMode— Ini adalah array peta pasangan kunci-nilai di mana:

  Setiap kunci adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Setiap nilai adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pengaturan Mode Lab yang digunakan oleh mesin.
- opencypher — Sebuah objek QueryLanguageVersion.

  Berisi informasi tentang bahasa query OpenCypher yang tersedia di cluster Anda. Secara khusus, ini berisi bidang versi yang menentukan versi OperCypher saat ini yang digunakan oleh mesin.
- role— String, tipe: `string` (string yang dikodekan UTF-8).

  Setel ke `reader` jika instance adalah read-replica, atau ke `writer` jika instance adalah instance utama.
- rollingBackTrxCount— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jika ada transaksi yang digulung kembali, bidang ini diatur ke jumlah transaksi tersebut. Jika tidak ada, bidang tidak muncul sama sekali.
- rollingBackTrxEarliestStartTime— String, tipe: `string` (string yang dikodekan UTF-8).

  Atur ke waktu mulai transaksi paling awal yang digulirkan kembali. Jika tidak ada transaksi yang diputar kembali, bidang tidak muncul sama sekali.

- settings— Ini adalah array peta pasangan kunci-nilai di mana:

  Setiap kunci adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Setiap nilai adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi informasi tentang pengaturan saat ini pada cluster DB Anda. Misalnya, berisi pengaturan batas waktu kueri klaster saat ini (`clusterQueryTimeoutInMs`).
- sparql — Sebuah objek QueryLanguageVersion.

  Berisi informasi tentang bahasa kueri SPARQL yang tersedia di klaster Anda. Secara khusus, ini berisi bidang versi yang menentukan versi SPARQL saat ini yang digunakan oleh mesin.
- startTime— String, tipe: `string` (string yang dikodekan UTF-8).

  Setel ke waktu UTC di mana proses server saat ini dimulai.
- status— String, tipe: `string` (string yang dikodekan UTF-8).

  Setel ke `healthy` jika instance tidak mengalami masalah. Jika instance pulih dari crash atau dari reboot dan ada transaksi aktif yang berjalan dari shutdown server terbaru, status diatur ke. `recovery`

Kesalahan

- UnsupportedOperationException
- InternalFailureException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- IllegalArgumentException
- TooManyRequestsException

ExecuteFastReset (tindakan)

Nama AWS CLI untuk API ini adalah:. `execute-fast-reset`

REST API reset cepat memungkinkan Anda mengatur ulang grafik Neptunus dengan cepat dan mudah, menghapus semua datanya.

Reset cepat Neptunus adalah proses dua langkah. Pertama Anda menelepon `ExecuteFastReset` dengan `action` set ke`initiateDatabaseReset`. Ini mengembalikan token UUID yang kemudian Anda sertakan saat memanggil `ExecuteFastReset` lagi dengan `action` set ke. `performDatabaseReset` Lihat Mengosongkan klaster DB Amazon Neptunus menggunakan API reset cepat.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan otentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. ResetDatabase

Permintaan

- action(dalam CLI:`--action`) - Diperlukan: Tindakan, tipe: `string` (string yang dikodekan UTF-8).

  Tindakan reset cepat. Salah satu nilai berikut:

  - **`initiateDatabaseReset`**— Tindakan ini menghasilkan token unik yang diperlukan untuk benar-benar melakukan reset cepat.
  - **`performDatabaseReset`**— Tindakan ini menggunakan token yang dihasilkan oleh `initiateDatabaseReset` tindakan untuk benar-benar melakukan reset cepat.

- token(dalam CLI:`--token`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Token reset cepat untuk memulai reset.

Respons

- payload — Sebuah objek FastResetToken.

  Hanya `payload` dikembalikan oleh `initiateDatabaseReset` tindakan, dan berisi token unik untuk digunakan dengan `performDatabaseReset` tindakan untuk membuat reset terjadi.

- status— Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Hanya `status` dikembalikan untuk `performDatabaseReset` tindakan, dan menunjukkan apakah rquest reset cepat diterima atau tidak.

Kesalahan

- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- ServerShutdownException
- PreconditionsFailedException
- MethodNotAllowedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Struktur operasi mesin:

QueryLanguageVersion (struktur)

Struktur untuk mengekspresikan versi bahasa kueri.

Bidang

- version— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Versi bahasa kueri.

FastResetToken (struktur)

Struktur yang berisi token reset cepat yang digunakan untuk memulai reset cepat.

Bidang

- token— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

Sebuah UUID dihasilkan oleh database dalam `initiateDatabaseReset` tindakan, dan kemudian dikonsumsi oleh `performDatabaseReset` untuk mengatur ulang database.

Kueri Neptunus APIs

Tindakan kueri Gremlin:

- ExecuteGremlinQuery (tindakan)
- ExecuteGremlinExplainQuery (tindakan)
- ExecuteGremlinProfileQuery (tindakan)
- ListGremlinQueries (tindakan)
- GetGremlinQueryStatus (tindakan)
- CancelGremlinQuery (tindakan)

Tindakan kueri OpenCypher:

- ExecuteOpenCypherQuery (tindakan)
- ExecuteOpenCypherExplainQuery (tindakan)
- ListOpenCypherQueries (tindakan)
- GetOpenCypherQueryStatus (tindakan)
- CancelOpenCypherQuery (tindakan)

Struktur kueri:

- QueryEvalStats (struktur)
- GremlinQueryStatus (struktur)
- GremlinQueryStatusAttributes (struktur)

ExecuteGremlinQuery (tindakan)

Nama AWS CLI untuk API ini adalah:. `execute-gremlin-query`

Perintah ini mengeksekusi query Gremlin. Amazon Neptunus kompatibel dengan TinkerPop Apache 3 dan Gremlin, sehingga Anda dapat menggunakan bahasa traversal Gremlin untuk menanyakan

grafik, seperti yang dijelaskan di bawah Grafik dalam dokumentasi Apache 3. TinkerPop Rincian lebih lanjut juga dapat ditemukan di Mengakses grafik Neptunus dengan Gremlin.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan salah satu tindakan IAM berikut di cluster itu, tergantung pada kueri:

- neptunus-db: ReadDataViaQuery
- neptunus-db: WriteDataViaQuery
- neptunus-db: DeleteDataViaQuery

Perhatikan bahwa kunci kondisi QueryLanguageIAM neptune-db ::Gremlin dapat digunakan dalam dokumen kebijakan untuk membatasi penggunaan kueri Gremlin (lihat Kunci kondisi yang tersedia di pernyataan kebijakan akses data IAM Neptunus).

Permintaan

- gremlinQuery(dalam CLI:`--gremlin-query`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Dengan menggunakan API ini, Anda dapat menjalankan kueri Gremlin dalam format string sebanyak yang Anda bisa menggunakan endpoint HTTP. Antarmuka kompatibel dengan versi Gremlin apa pun yang digunakan cluster DB Anda (lihat bagian klien Tinkerpop untuk menentukan Gremlin mana yang merilis versi mesin Anda yang didukung).
- serializer(dalam CLI:`--serializer`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jika non-null, hasil query dikembalikan dalam pesan respon serial dalam format yang ditentukan oleh parameter ini. Lihat bagian GraphSon dalam TinkerPop dokumentasi untuk daftar format yang saat ini didukung.

Respons

- meta— Dokumen, tipe: `document` (konten terbuka protokol-agnostik yang diwakili oleh model data seperti JSON).

  Metadata tentang kueri Gremlin.
- requestId— String, tipe: `string` (string yang dikodekan UTF-8).

Pengidentifikasi unik dari kueri Gremlin.

- result— Dokumen, tipe: `document` (konten terbuka protokol-agnostik yang diwakili oleh model data seperti JSON).

  Output query Gremlin dari server.

- status — Sebuah objek GremlinQueryStatusAttributes.

  Status kueri Gremlin.

Kesalahan

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteGremlinExplainQuery (tindakan)

Nama AWS CLI untuk API ini adalah:. `execute-gremlin-explain-query`

Mengeksekusi kueri Gremlin Expline.

Amazon Neptunus telah menambahkan fitur Gremlin `explain` bernama yang menyediakan alat swalayan untuk memahami pendekatan eksekusi yang diambil oleh mesin Neptunus untuk kueri. Anda memintanya dengan menambahkan parameter `explain` ke panggilan HTTP yang mengirimkan kueri Gremlin.

Fitur menjelaskan memberikan informasi tentang struktur logis dari rencana eksekusi kueri. Anda dapat menggunakan informasi ini untuk mengidentifikasi potensi kemacetan evaluasi dan eksekusi dan untuk menyetel kueri Anda, seperti yang dijelaskan dalam kueri Tuning Gremlin. Anda juga dapat menggunakan petunjuk kueri untuk meningkatkan rencana eksekusi kueri.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan otentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan salah satu tindakan IAM berikut di cluster itu, tergantung pada kueri:

- neptunus-db: ReadDataViaQuery
- neptunus-db: WriteDataViaQuery
- neptunus-db: DeleteDataViaQuery

Perhatikan bahwa kunci kondisi QueryLanguageIAM neptune-db ::Gremlin dapat digunakan dalam dokumen kebijakan untuk membatasi penggunaan kueri Gremlin (lihat Kunci kondisi yang tersedia di pernyataan kebijakan akses data IAM Neptunus).

Permintaan

- gremlinQuery(dalam CLI:`--gremlin-query`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Gremlin menjelaskan string kueri.

Respons

- output— a ReportAsText, tipe: `blob` (blok data biner yang tidak ditafsirkan).

Gumpalan teks yang berisi Gremlin menjelaskan hasil, seperti yang dijelaskan dalam kueri Tuning Gremlin.

Kesalahan

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteGremlinProfileQuery (tindakan)

Nama AWS CLI untuk API ini adalah:. `execute-gremlin-profile-query`

Mengeksekusi kueri Profil Gremlin, yang menjalankan traversal tertentu, mengumpulkan berbagai metrik tentang proses, dan menghasilkan laporan profil sebagai output. Lihat API profil Gremlin di Neptunus untuk detailnya.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. ReadDataViaQuery

Perhatikan bahwa kunci kondisi QueryLanguageIAM neptune-db ::Gremlin dapat digunakan dalam dokumen kebijakan untuk membatasi penggunaan kueri Gremlin (lihat Kunci kondisi yang tersedia di pernyataan kebijakan akses data IAM Neptunus).

Permintaan

- chop(dalam CLI:`--chop`) — Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jika non-nol, menyebabkan hasil string akan dipotong pada jumlah karakter tersebut. Jika diatur ke nol, string berisi semua hasil.
- gremlinQuery(dalam CLI:`--gremlin-query`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  String kueri Gremlin ke profil.
- indexOps(dalam CLI:`--index-ops`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika tanda ini disetel ke`TRUE`, hasilnya menyertakan laporan terperinci dari semua operasi indeks yang terjadi selama eksekusi kueri dan serialisasi.
- results(dalam CLI:`--results`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika tanda ini disetel ke`TRUE`, hasil kueri dikumpulkan dan ditampilkan sebagai bagian dari laporan profil. Jika`FALSE`, hanya jumlah hasil yang ditampilkan.
- serializer(dalam CLI:`--serializer`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jika non-null, hasil yang dikumpulkan dikembalikan dalam pesan respons serial dalam format yang ditentukan oleh parameter ini. Lihat API profil Gremlin di Neptunus untuk informasi selengkapnya.

Respons

- output— a ReportAsText, tipe: `blob` (blok data biner yang tidak ditafsirkan).

Gumpalan teks yang berisi hasil Profil Gremlin. Lihat API profil Gremlin di Neptunus untuk detailnya.

Kesalahan

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ListGremlinQueries (tindakan)

Nama AWS CLI untuk API ini adalah:. `list-gremlin-queries`

Daftar kueri Gremlin aktif. Lihat API status kueri Gremlin untuk detail tentang output.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. GetQueryStatus

Perhatikan bahwa kunci kondisi QueryLanguageIAM neptune-db ::Gremlin dapat digunakan dalam dokumen kebijakan untuk membatasi penggunaan kueri Gremlin (lihat Kunci kondisi yang tersedia di pernyataan kebijakan akses data IAM Neptunus).

Permintaan

- includeWaiting(dalam CLI:`--include-waiting`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke`TRUE`, daftar yang dikembalikan menyertakan kueri tunggu. Defaultnya adalah`FALSE`;

Respons

- acceptedQueryCount— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah kueri yang telah diterima tetapi belum selesai, termasuk kueri dalam antrian.
- queries – Susunan objek GremlinQueryStatus.

  Daftar kueri saat ini.
- runningQueryCount— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah kueri Gremlin yang sedang berjalan.

Kesalahan

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException

- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

GetGremlinQueryStatus (tindakan)

Nama AWS CLI untuk API ini adalah:. `get-gremlin-query-status`

Mendapat status query Gremlin tertentu.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. GetQueryStatus

Perhatikan bahwa kunci kondisi QueryLanguageIAM neptune-db ::Gremlin dapat digunakan dalam dokumen kebijakan untuk membatasi penggunaan kueri Gremlin (lihat Kunci kondisi yang tersedia di pernyataan kebijakan akses data IAM Neptunus).

Permintaan

- queryId(dalam CLI:`--query-id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik yang mengidentifikasi kueri Gremlin.

Respons

- queryEvalStats — Sebuah objek QueryEvalStats.

  Status evaluasi kueri Gremlin.
- queryId— String, tipe: `string` (string yang dikodekan UTF-8).

ID kueri yang statusnya dikembalikan.

- queryString— String, tipe: `string` (string yang dikodekan UTF-8).

  String kueri Gremlin.

Kesalahan

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

CancelGremlinQuery (tindakan)

Nama AWS CLI untuk API ini adalah:. `cancel-gremlin-query`

Membatalkan kueri Gremlin. Lihat pembatalan kueri Gremlin untuk informasi selengkapnya.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. CancelQuery

Permintaan

- queryId(dalam CLI:`--query-id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik yang mengidentifikasi kueri yang akan dibatalkan.

Respons

- status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status pembatalan

Kesalahan

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

Tindakan kueri OpenCypher:

ExecuteOpenCypherQuery (tindakan)

Nama AWS CLI untuk API ini adalah:. `execute-open-cypher-query`

Mengeksekusi query OpenCypher. Lihat Mengakses Grafik Neptunus dengan OpenCypher untuk informasi lebih lanjut.

Neptunus mendukung pembuatan aplikasi grafik menggunakan OpenCypher, yang saat ini merupakan salah satu bahasa kueri paling populer di kalangan pengembang yang bekerja dengan database grafik. Pengembang, analis bisnis, dan ilmuwan data menyukai sintaks deklaratif dan terinspirasi SQL OpenCypher karena menyediakan struktur yang akrab untuk menanyakan grafik properti.

Bahasa OpenCypher awalnya dikembangkan oleh Neo4j, kemudian bersumber terbuka pada tahun 2015 dan berkontribusi pada proyek OpenCypher di bawah lisensi open-source Apache 2.

Perhatikan bahwa saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan salah satu tindakan IAM berikut di klaster tersebut, tergantung pada kueri:

- neptunus-db: ReadDataViaQuery
- neptunus-db: WriteDataViaQuery
- neptunus-db: DeleteDataViaQuery

Perhatikan juga bahwa kunci kondisi OpenCypher IAM neptune-db:QueryLanguage: dapat digunakan dalam dokumen kebijakan untuk membatasi penggunaan kueri OpenCypher (lihat Kunci kondisi yang tersedia dalam pernyataan kebijakan akses data IAM Neptunus).

Permintaan

- openCypherQuery(dalam CLI:`--open-cypher-query`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  String query OpenCypher yang akan dieksekusi.
- parameters(dalam CLI:`--parameters`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Parameter query OpenCypher untuk eksekusi query. Lihat Contoh kueri parameter OpenCypher untuk informasi selengkapnya.

Respons

- results— Diperlukan: Dokumen, tipe: `document` (konten terbuka protokol-agnostik yang diwakili oleh model data seperti JSON).

  Hasil OpenCypherQuery.

Kesalahan

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteOpenCypherExplainQuery (tindakan)

Nama AWS CLI untuk API ini adalah:. `execute-open-cypher-explain-query`

Mengeksekusi permintaan OpenCypher`explain`. Lihat fitur penjelasan OpenCypher untuk informasi selengkapnya.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. ReadDataViaQuery

Perhatikan bahwa kunci kondisi OpenCypher IAM neptune-db:QueryLanguage: dapat digunakan dalam dokumen kebijakan untuk membatasi penggunaan kueri OpenCypher (lihat Kunci kondisi yang tersedia dalam pernyataan kebijakan akses data IAM Neptunus).

Permintaan

- explainMode(dalam CLI:`--explain-mode`) - Wajib: an OpenCypherExplainMode, tipe: `string` (string yang dikodekan UTF-8).

  Mode OpenCypher`explain`. Bisa menjadi salah satu dari:`static`,`dynamic`, atau`details`.
- openCypherQuery(dalam CLI:`--open-cypher-query`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  String query OpenCypher.
- parameters(dalam CLI:`--parameters`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Parameter kueri OpenCypher.

Respons

- results— Diperlukan: Gumpalan, tipe: `blob` (blok data biner yang tidak ditafsirkan).

  Sebuah gumpalan teks yang berisi hasil OpenCypher`explain`.

Kesalahan

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException

- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ListOpenCypherQueries (tindakan)

> Nama AWS CLI untuk API ini adalah:. `list-open-cypher-queries`

Daftar query OpenCypher aktif. Lihat titik akhir status Neptunus OpenCypher untuk informasi lebih lanjut.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. GetQueryStatus

Perhatikan bahwa kunci kondisi OpenCypher IAM neptune-db:QueryLanguage: dapat digunakan dalam dokumen kebijakan untuk membatasi penggunaan kueri OpenCypher (lihat Kunci kondisi yang tersedia dalam pernyataan kebijakan akses data IAM Neptunus).

Permintaan

- includeWaiting(dalam CLI:`--include-waiting`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Ketika disetel ke TRUE dan parameter lain tidak ada, menyebabkan informasi status dikembalikan untuk kueri menunggu serta untuk menjalankan kueri.

Respons

- acceptedQueryCount— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah kueri yang telah diterima tetapi belum selesai, termasuk kueri dalam antrian.
- queries – Susunan objek GremlinQueryStatus.

  Daftar kueri OpenCypher saat ini.
- runningQueryCount— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah kueri OpenCypher yang sedang berjalan.

Kesalahan

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException

- MissingParameterException
- ConcurrentModificationException

GetOpenCypherQueryStatus (tindakan)

Nama AWS CLI untuk API ini adalah:. `get-open-cypher-query-status`

Mengambil status query OpenCypher tertentu.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. GetQueryStatus

Perhatikan bahwa kunci kondisi OpenCypher IAM neptune-db:QueryLanguage: dapat digunakan dalam dokumen kebijakan untuk membatasi penggunaan kueri OpenCypher (lihat Kunci kondisi yang tersedia dalam pernyataan kebijakan akses data IAM Neptunus).

Permintaan

- queryId(dalam CLI:`--query-id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID unik dari query OpenCypher untuk mengambil status query.

Respons

- queryEvalStats — Sebuah objek QueryEvalStats.

  Status evaluasi kueri OpenCypher.
- queryId— String, tipe: `string` (string yang dikodekan UTF-8).

  ID unik dari kueri yang statusnya dikembalikan.
- queryString— String, tipe: `string` (string yang dikodekan UTF-8).

  String query OpenCypher.

Kesalahan

- InvalidNumericDataException

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

CancelOpenCypherQuery (tindakan)

Nama AWS CLI untuk API ini adalah:. `cancel-open-cypher-query`

Membatalkan query OpenCypher tertentu. Lihat titik akhir status Neptunus OpenCypher untuk informasi lebih lanjut.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. CancelQuery

Permintaan

- queryId(dalam CLI:`--query-id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID unik dari kueri OpenCypher untuk dibatalkan.
- silent(dalam CLI:`--silent`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

Jika disetel keTRUE, menyebabkan pembatalan kueri OpenCypher terjadi secara diam-diam.

Respons

- payload— Boolean, tipe: `boolean` (nilai Boolean (benar atau salah)).

  Payload pembatalan untuk kueri OpenCypher.
- status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status pembatalan kueri OpenCypher.

Kesalahan

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

Struktur kueri:

QueryEvalStats (struktur)

Struktur untuk menangkap statistik kueri seperti berapa banyak kueri yang berjalan, diterima atau menunggu dan detailnya.

Bidang

- cancelled— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Setel ke TRUE jika kueri dibatalkan, atau FALSE sebaliknya.
- elapsed— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah milidetik kueri telah berjalan sejauh ini.
- subqueries— Ini adalah Dokumen, dari tipe: `document` (konten terbuka protokol-agnostik yang diwakili oleh model data seperti JSON).

  Jumlah subquery dalam query ini.
- waited— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Menunjukkan berapa lama kueri menunggu, dalam milidetik.

GremlinQueryStatus (struktur)

Menangkap status kueri Gremlin (lihat halaman API status kueri Gremlin).

Bidang

- queryEvalStatsIni adalah sebuah QueryEvalStats objek.

  Statistik kueri dari kueri Gremlin.
- queryId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  ID dari kueri Gremlin.
- queryString— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  String query dari query Gremlin.

GremlinQueryStatusAttributes (struktur)

Berisi komponen status dari query Gremlin.

Bidang

- attributes— Ini adalah Dokumen, dari tipe: `document` (konten terbuka protokol-agnostik yang diwakili oleh model data seperti JSON).

  Atribut status kueri Gremlin.
- code— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Kode respons HTTP dikembalikan dari permintaan kueri Gremlin..
- message— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan status.

Pemuat massal pesawat data Neptunus APIs

Tindakan beban massal:

- StartLoaderJob (tindakan)
- GetLoaderJobStatus (tindakan)
- ListLoaderJobs (tindakan)
- CancelLoaderJob (tindakan)

Struktur beban massal:

- LoaderIdResult (struktur)

StartLoaderJob (tindakan)

Nama AWS CLI untuk API ini adalah:. `start-loader-job`

Memulai tugas pemuat massal Neptunus untuk memuat data dari bucket Amazon S3 ke instans DB Neptunus. Lihat Menggunakan Amazon Neptune Bulk Loader untuk Menyerap Data.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. StartLoaderJob

Permintaan

- dependencies(dalam CLI:`--dependencies`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Ini adalah parameter opsional yang dapat membuat permintaan beban antrian bergantung pada keberhasilan penyelesaian satu atau lebih pekerjaan sebelumnya dalam antrian.

  Neptune dapat mengantrekan sebanyak 64 permintaan pemuatan sekaligus, jika parameter `queueRequest` permintaannya diatur ke `"TRUE"`. Parameter `dependencies` memungkinkan Anda melakukan eksekusi seperti permintaan mengantre yang tergantung pada penyelesaian yang berhasil dari satu atau lebih permintaan ditentukan sebelumnya dalam antrean.

  Misalnya, jika pemuatan `Job-A` dan `Job-B` independen satu sama lain, namun pemuatan `Job-C` membutuhkan `Job-A` dan `Job-B` harus selesai sebelum dimulai, lanjutkan sebagai berikut:

  1. Kirim `load-job-A` dan `load-job-B` satu demi satu dalam urutan apa pun, dan simpan load-id mereka.
  2. Kirim `load-job-C` dengan load-id dari dua pekerjaan di bidang `dependencies`-nya:

  Example

  ```
  "dependencies" : ["(job_A_load_id)", "(job_B_load_id)"]
  ```

  Karena parameter `dependencies`, loader massal tidak akan memulai `Job-C` sampai `Job-A` dan `Job-B` telah berhasil diselesaikan. Jika salah satu dari mereka gagal, Job-C tidak akan dieksekusi, dan statusnya akan diatur ke `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

  Anda dapat mengatur beberapa tingkat ketergantungan dengan cara ini, sehingga kegagalan satu pekerjaan akan menyebabkan semua permintaan yang secara langsung atau tidak langsung tergantung padanya untuk dibatalkan.

- failOnError(dalam CLI:`--fail-on-error`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  `failOnError` — Sebuah bendera untuk mengubah berhenti penuh pada kesalahan.

Nilai yang diizinkan: `"TRUE"`, `"FALSE"`.

Nilai default:`"TRUE"`.

Ketika parameter ini diatur ke `"FALSE"`, loader mencoba memuat semua data di lokasi yang ditentukan, melewatkan entri apa pun yang memiliki kesalahan.

Ketika parameter ini diatur ke `"TRUE"`, loader berhenti segera setelah menemukan kesalahan. Data yang dimuat sampai saat itu tetap ada.

- format(dalam CLI:`--format`) - Diperlukan: Format, tipe: `string` (string yang dikodekan UTF-8).

  Format data. Untuk informasi selengkapnya tentang format data untuk perintah `Loader` Neptunus, lihat Memuat Format Data.

  Nilai yang diizinkan

  - **`csv`**untuk format data CSV Gremlin.
  - **`opencypher`**untuk format data CSV OpenCypher.
  - **`ntriples`**untuk format data RDF N-Triples.
  - **`nquads`**untuk format data N-Quads RDF.
  - **`rdfxml`**untuk format data RDF\ XHTML RDF.
  - **`turtle`**untuk format data Turtle RDF.
- iamRoleArn(dalam CLI:`--iam-role-arn`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) untuk peran IAM akan diasumsikan oleh instans DB Neptunus untuk akses ke bucket S3. Peran IAM ARN yang disediakan di sini harus dilampirkan ke cluster DB (lihat Menambahkan Peran IAM ke Cluster Amazon Neptunus.
- mode(dalam CLI:`--mode`) — Mode, tipe: `string` (string yang dikodekan UTF-8).

  Mode beban pekerjaan.

  Nilai yang diizinkan:`RESUME`,`NEW`,`AUTO`.

  Nilai default:`AUTO`.

- `RESUME` — Dalam mode RESUME, loader mencari pemuatan sebelumnya dari sumber ini, dan jika menemukan satu, melanjutkan pekerjaan pemuatan tersebut. Jika tidak ada pekerjaan pemuatan sebelumnya yang ditemukan, loader berhenti.

  Loader menghindari memuat ulang file yang berhasil dimuat di pekerjaan sebelumnya. Ia hanya mencoba untuk memproses file yang gagal. Jika Anda menjatuhkan data yang dimuat sebelumnya dari klaster Neptune Anda, data tersebut tidak dimuat ulang dalam mode ini. Jika pekerjaan pemuatan sebelumnya berhasil memuat semua file dari sumber yang sama, tidak ada yang dimuat ulang, dan loader mengembalikan keberhasilan.
- `NEW` — Dalam mode NEW, it menciptakan permintaan pemuatan baru terlepas dari pemuatan sebelumnya. Anda dapat menggunakan mode ini untuk memuat ulang semua data dari sumber setelah menjatuhkan data yang dimuat sebelumnya dari klaster Neptune Anda, atau untuk memuat data baru yang tersedia di sumber yang sama.
- `AUTO` — Dalam mode AUTO, loader mencari pekerjaan pemuatan sebelumnya dari sumber yang sama, dan jika menemukannya, melanjutkan pekerjaan itu, seperti pada mode `RESUME`.

  Jika loader tidak menemukan pekerjaan pemuatan sebelumnya dari sumber yang sama, loader akan memuat semua data dari sumbernya, seperti pada mode `NEW`.

- parallelism(dalam CLI:`--parallelism`) — Paralelisme, tipe: `string` (string yang dikodekan UTF-8).

  `parallelism`Parameter opsional dapat diatur untuk mengurangi jumlah utas yang digunakan oleh proses beban massal.

  Nilai yang diizinkan:

  - `LOW`— Jumlah benang yang digunakan adalah jumlah v yang tersedia CPUs dibagi 8.
  - `MEDIUM`— Jumlah benang yang digunakan adalah jumlah v yang tersedia CPUs dibagi 2.
  - `HIGH`— Jumlah thread yang digunakan sama dengan jumlah yang tersedia vCPUs.
  - `OVERSUBSCRIBE`— Jumlah utas yang digunakan adalah jumlah v yang tersedia CPUs dikalikan dengan 2. Jika nilai ini digunakan, loader massal mengambil semua sumber daya yang tersedia.

    Ini tidak berarti, bagaimanapun, bahwa pengaturan `OVERSUBSCRIBE` menghasilkan 100% utilisasi CPU. Karena operasi beban I/O terikat, pemanfaatan CPU tertinggi yang diharapkan adalah dalam kisaran 60% hingga 70%.

  Nilai default: `HIGH`

`parallelism`Pengaturan terkadang dapat mengakibatkan kebuntuan antar utas saat memuat data OpenCypher. Ketika ini terjadi, Neptunus mengembalikan kesalahan. `LOAD_DATA_DEADLOCK` Anda biasanya dapat memperbaiki masalah dengan menyetel `parallelism` ke pengaturan yang lebih rendah dan mencoba kembali perintah load.

- parserConfiguration(dalam CLI:`--parser-configuration`) — Ini adalah array peta pasangan kunci-nilai di mana:

  Setiap kunci adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Setiap nilai adalah String, tipe: `string` (string yang dikodekan UTF-8).

  `parserConfiguration` — Sebuah objek opsional dengan nilai konfigurasi parser tambahan. Masing-masing parameter turunan juga opsional:

  - **`namedGraphUri`**— Grafik default untuk semua format RDF ketika tidak ada grafik yang ditentukan (untuk format non-quads dan entri NQUAD tanpa grafik).

    Nilai default-nya `https://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

  - **`baseUri`**— URI dasar untuk RDF/XML dan format Turtle.

    Nilai default-nya `https://aws.amazon.com/neptune/default`.

  - **`allowEmptyStrings`**— Pengguna Gremlin harus dapat memberikan nilai string kosong (“”) sebagai properti simpul dan tepi saat memuat data CSV. Jika `allowEmptyStrings` diatur ke `false` (default), string kosong diperlakukan sebagai null dan tidak dimuat.

    Jika `allowEmptyStrings` diatur ke `true`, loader memperlakukan string kosong sebagai nilai properti yang valid dan memuatnya sesuai keperluan.

- queueRequest(dalam CLI:`--queue-request`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Ini adalah parameter bendera opsional yang menunjukkan apakah permintaan pemuatan dapat diantrian atau tidak.

  Anda tidak perlu menunggu satu pekerjaan muat selesai sebelum mengeluarkan pekerjaan berikutnya, karena Neptune dapat mengantrekan sebanyak 64 pekerjaan sekaligus, asalkan parameter `queueRequest` semua diatur ke `"TRUE"`. Urutan antrian pekerjaan akan menjadi first-in-first-out (FIFO).

Jika parameter `queueRequest` dihilangkan atau diatur ke `"FALSE"`, permintaan pemuatan akan gagal jika pekerjaan pemuatan lain sudah berjalan.

Nilai yang diizinkan: `"TRUE"`, `"FALSE"`.

Nilai default:`"FALSE"`.

- s3BucketRegion(dalam CLI:`--s-3-bucket-region`) - Diperlukan: S3BucketRegion, tipe: `string` (string yang dikodekan UTF-8).

  Wilayah Amazon dari ember S3. Ini harus cocok dengan Wilayah Amazon dari cluster DB.

- source(dalam CLI:`--source`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  `source`Parameter menerima URI S3 yang mengidentifikasi satu file, beberapa file, folder, atau beberapa folder. Neptune memuat setiap file data dalam folder yang ditentukan.

  URI dapat berupa format berikut.

  - `s3://(bucket_name)/(object-key-name)`
  - `https://s3.amazonaws.com/(bucket_name)/(object-key-name)`
  - `https://s3.us-east-1.amazonaws.com/(bucket_name)/(object-key-name)`

  `object-key-name`Elemen URI setara dengan parameter awalan dalam panggilan ListObjectsAPI S3. Ini mengidentifikasi semua objek dalam ember S3 yang ditentukan yang namanya dimulai dengan awalan itu. Objek itu bisa berupa satu file atau folder, atau beberapa file dan/atau folder.

  Folder atau folder-folder yang ditentukan dapat berisi beberapa file vertex dan beberapa file edge.

- updateSingleCardinalityProperties(dalam CLI:`--update-single-cardinality-properties`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  `updateSingleCardinalityProperties`adalah parameter opsional yang mengontrol cara pemuat massal memperlakukan nilai baru untuk properti simpul atau tepi kardinalitas tunggal. Ini tidak didukung untuk memuat data OpenCypher.

  Nilai yang diizinkan: `"TRUE"`, `"FALSE"`.

  Nilai default:`"FALSE"`.

Secara default, atau saat `updateSingleCardinalityProperties` secara eksplisit diatur ke `"FALSE"`, loader memperlakukan nilai baru sebagai kesalahan, karena melanggar kardinalitas tunggal.

Saat `updateSingleCardinalityProperties` diatur ke `"TRUE"`, di sisi lain, loader massal menggantikan nilai yang ada dengan yang baru. Jika beberapa edge atau nilai properti vertex single-cardinality disediakan dalam file sumber yang dimuat, nilai akhir pada akhir pemuatan massal bisa menjadi salah satu dari nilai-nilai baru tersebut. Loader hanya menjamin bahwa nilai yang ada telah digantikan oleh salah satu yang baru.

- userProvidedEdgeIds(dalam CLI:`--user-provided-edge-ids`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Parameter ini diperlukan hanya ketika memuat data OpenCypher yang berisi hubungan. IDs Itu harus disertakan dan diatur ke `True` ketika hubungan OpenCypher secara eksplisit IDs disediakan dalam data beban (disarankan).

  Ketika tidak `userProvidedEdgeIds` ada atau diatur ke`True`, `:ID` kolom harus ada di setiap file hubungan dalam beban.

  Ketika `userProvidedEdgeIds` hadir dan diatur ke`False`, file hubungan dalam beban tidak boleh berisi `:ID` kolom. Sebagai gantinya, pemuat Neptunus secara otomatis menghasilkan ID untuk setiap hubungan.

  Sangat berguna untuk memberikan hubungan secara IDs eksplisit sehingga loader dapat melanjutkan pemuatan setelah kesalahan dalam data CSV telah diperbaiki, tanpa harus memuat ulang hubungan apa pun yang telah dimuat. Jika hubungan IDs belum ditetapkan secara eksplisit, pemuat tidak dapat melanjutkan pemuatan yang gagal jika ada file hubungan yang harus diperbaiki, dan sebagai gantinya harus memuat ulang semua hubungan.

Respons

- payload— Wajib: Ini adalah array peta pasangan kunci-nilai di mana:

  Setiap kunci adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Setiap nilai adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Berisi pasangan `loadId` nama-nilai yang menyediakan pengenal untuk operasi beban.

- status— Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode pengembalian HTTP yang menunjukkan status pekerjaan pemuatan.

Kesalahan

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- S3Exception
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

GetLoaderJobStatus (tindakan)

Nama AWS CLI untuk API ini adalah:. `get-loader-job-status`

Mendapat informasi status tentang pekerjaan pemuatan tertentu. Neptunus melacak 1.024 pekerjaan pemuatan massal terbaru, dan menyimpan 10.000 detail kesalahan terakhir per pekerjaan.

Lihat Neptune Get-Status Loader API untuk informasi selengkapnya.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan otentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu.. GetLoaderJobStatus

Permintaan

- details(dalam CLI:`--details`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Tandai yang menunjukkan apakah akan menyertakan detail di luar status keseluruhan atau tidak (`TRUE`atau`FALSE`; defaultnya adalah`FALSE`).
- errors(dalam CLI:`--errors`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Bendera yang menunjukkan apakah akan menyertakan daftar kesalahan yang ditemui atau tidak (`TRUE`atau`FALSE`; defaultnya adalah`FALSE`).

  Daftar kesalahan dipecah dalam beberapa bagian. Parameter `page` dan `errorsPerPage` memungkinkan Anda untuk melalui semua kesalahan dalam beberapa bagian.
- errorsPerPage(dalam CLI:`--errors-per-page`) — a PositiveInteger, tipe: `integer` (bilangan bulat 32-bit yang ditandatangani), setidaknya 1? st?.

  Jumlah kesalahan yang dikembalikan di setiap halaman (bilangan bulat positif; defaultnya adalah`10`). Hanya berlaku ketika `errors` parameter disetel ke`TRUE`.
- loadId(dalam CLI:`--load-id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID beban pekerjaan beban untuk mendapatkan status.
- page(dalam CLI:`--page`) — a PositiveInteger, tipe: `integer` (bilangan bulat 32-bit yang ditandatangani), setidaknya 1? st?.

  Nomor halaman kesalahan (bilangan bulat positif; defaultnya adalah`1`). Hanya berlaku ketika `errors` parameter disetel ke`TRUE`.

Respons

- payload— Diperlukan: Dokumen, tipe: `document` (konten terbuka protokol-agnostik yang diwakili oleh model data seperti JSON).

  Informasi status tentang pekerjaan pemuatan, dalam tata letak yang bisa terlihat seperti ini:

  Example

```
{
          "status" : "200 OK",
          "payload" : {
            "feedCount" : [
              {
```

```
            "LOAD_FAILED" : (number)
          }
        ],
        "overallStatus" : {
          "fullUri" : "s3://(bucket)/(key)",
          "runNumber" : (number),
          "retryNumber" : (number),
          "status" : "(string)",
          "totalTimeSpent" : (number),
          "startTime" : (number),
          "totalRecords" : (number),
          "totalDuplicates" : (number),
          "parsingErrors" : (number),
          "datatypeMismatchErrors" : (number),
          "insertErrors" : (number),
        },
        "failedFeeds" : [
          {
            "fullUri" : "s3://(bucket)/(key)",
            "runNumber" : (number),
            "retryNumber" : (number),
            "status" : "(string)",
            "totalTimeSpent" : (number),
            "startTime" : (number),
            "totalRecords" : (number),
            "totalDuplicates" : (number),
            "parsingErrors" : (number),
            "datatypeMismatchErrors" : (number),
            "insertErrors" : (number),
          }
        ],
        "errors" : {
          "startIndex" : (number),
          "endIndex" : (number),
          "loadId" : "(string),
          "errorLogs" : [ ]
        }
      }
    }
```

- status— Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode respons HTTP untuk permintaan tersebut.

Kesalahan

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ListLoaderJobs (tindakan)

Nama AWS CLI untuk API ini adalah:. `list-loader-jobs`

Mengambil daftar `loadIds` untuk semua pekerjaan loader aktif.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan otentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu.. ListLoaderJobs

Permintaan

- includeQueuedLoads(dalam CLI:`--include-queued-loads`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Parameter opsional yang dapat digunakan untuk mengecualikan beban IDs permintaan beban antrian saat meminta daftar beban IDs dengan menyetel parameter ke. `FALSE` Nilai default-nya adalah `TRUE`.
- limit(dalam CLI:`--limit`) — a ListLoaderJobsInputLimitInteger, tipe: `integer` (bilangan bulat 32-bit yang ditandatangani), tidak kurang dari 1 atau lebih dari 100? st? s.

Jumlah beban IDs ke daftar. Harus bilangan bulat positif lebih besar dari nol dan tidak lebih dari `100` (yang merupakan default).

Respons

- payload — Wajib: Sebuah objek LoaderIdResult.

  Daftar pekerjaan yang diminta IDs.
- status— Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Mengembalikan status permintaan daftar pekerjaan.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- InternalFailureException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException

CancelLoaderJob (tindakan)

Nama AWS CLI untuk API ini adalah:. `cancel-loader-job`

Membatalkan pekerjaan pemuatan tertentu. Ini adalah `DELETE` permintaan HTTP. Lihat Neptune Get-Status Loader API untuk informasi selengkapnya.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan otentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu.. CancelLoaderJob

Permintaan

- loadId(dalam CLI:`--load-id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID pekerjaan pemuatan yang akan dihapus.

Respons

- status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status pembatalan.

Kesalahan

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Struktur beban massal:

LoaderIdResult (struktur)

Berisi daftar beban IDs.

Bidang

- loadIds— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar beban IDs.

Neptunus mengalirkan API jalur data

Tindakan akses streaming:

- GetPropertygraphStream (tindakan)

Struktur data aliran:

- PropertygraphRecord (struktur)
- PropertygraphData (struktur)

GetPropertygraphStream (tindakan)

Nama AWS CLI untuk API ini adalah:. `get-propertygraph-stream`

Mendapat aliran untuk grafik properti.

Dengan fitur Neptunus Streams, Anda dapat menghasilkan urutan lengkap entri log perubahan yang merekam setiap perubahan yang dilakukan pada data grafik Anda saat itu terjadi. `GetPropertygraphStream`memungkinkan Anda mengumpulkan entri log perubahan ini untuk grafik properti.

Fitur aliran Neptunus perlu diaktifkan di Neptunus Anda. DBcluster Untuk mengaktifkan aliran, setel parameter cluster DB neptune_streams ke. 1

Lihat Menangkap perubahan grafik secara real time menggunakan aliran Neptunus.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. GetStreamRecords

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan salah satu tindakan IAM berikut, tergantung pada kueri:

Perhatikan bahwa Anda dapat membatasi kueri grafik properti menggunakan kunci konteks IAM berikut:

- neptunus-db ::Gremlin QueryLanguage
- neptunus-db:: QueryLanguage OpenCypher

Lihat Kunci kondisi yang tersedia di pernyataan kebijakan akses data IAM Neptunus).

Permintaan

- commitNum(dalam CLI:`--commit-num`) — Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Jumlah commit dari catatan awal yang akan dibaca dari stream log perubahan. Parameter ini `iteratorType` diperlukan saat `AT_SEQUENCE_NUMBER` atau`AFTER_SEQUENCE_NUMBER`, dan diabaikan kapan `iteratorType` `TRIM_HORIZON` atau`LATEST`.
- encoding(dalam CLI:`--encoding`) — Pengkodean, tipe: `string` (string yang dikodekan UTF-8).

  Jika disetel ke TRUE, Neptunus memampatkan respons menggunakan pengkodean gzip.
- iteratorType(dalam CLI:`--iterator-type`) — sebuah IteratorType, dari tipe: `string` (string yang dikodekan UTF-8).

  Bisa menjadi salah satu dari:
  - `AT_SEQUENCE_NUMBER`— Menunjukkan bahwa pembacaan harus dimulai dari nomor urut acara yang ditentukan bersama oleh opNum parameter `commitNum` dan.
  - `AFTER_SEQUENCE_NUMBER`— Menunjukkan bahwa pembacaan harus dimulai tepat setelah nomor urut acara yang ditentukan bersama oleh opNum parameter `commitNum` dan.
  - `TRIM_HORIZON`— Menunjukkan bahwa pembacaan harus dimulai pada catatan terakhir yang belum dipangkas dalam sistem, yang merupakan catatan tertua yang belum kedaluwarsa (belum dihapus) dalam aliran log perubahan.

  - `LATEST`— Menunjukkan bahwa pembacaan harus dimulai pada catatan terbaru dalam sistem, yang merupakan catatan terbaru yang belum kedaluwarsa (belum dihapus) dalam aliran log perubahan.
- limit(dalam CLI:`--limit`) — a GetPropertygraphStreamInputLimitLong, tipe: `long` (integer 64-bit yang ditandatangani), tidak kurang dari 1 atau lebih dari 100000? st? s.

  Menentukan angka maksimum catatan yang akan dikembalikan. Ada juga batas ukuran 10 MB pada respon yang tidak dapat diubah dan yang diutamakan daripada jumlah catatan yang ditentukan dalam parameter `limit`. Respon mencakup catatan pelanggaran ambang batas jika batas 10 MB tercapai.

  Rentang untuk `limit` adalah 1 hingga 100.000, dengan default 10.
- opNum(dalam CLI:`--op-num`) — Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Nomor urut operasi dalam commit yang ditentukan untuk mulai dibaca dari dalam data stream log perubahan. Nilai default-nya 1.

Respons

- format— Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Format serialisasi untuk catatan perubahan yang dikembalikan. Saat ini, satu-satunya nilai yang didukung adalah `PG_JSON`.
- lastEventId— Wajib: Ini adalah array peta pasangan kunci-nilai di mana:

    Setiap kunci adalah String, tipe: `string` (string yang dikodekan UTF-8).

    Setiap nilai adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi urutan dari perubahan terakhir dalam respons aliran.

  ID peristiwa terdiri dari dua bidang: `acommitNum`, yang mengidentifikasi transaksi yang mengubah grafik, dan`opNum`, yang mengidentifikasi operasi tertentu dalam transaksi itu:

  Example

```
"eventId": {
          "commitNum": 12,
          "opNum": 1
```

```
        }
```

- lastTrxTimestampInMillis— Diperlukan: Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Waktu di mana komit untuk transaksi diminta, dalam milidetik dari zaman Unix.

- records – Wajib: Susunan objek PropertygraphRecord.

  Array catatan aliran log perubahan serial yang disertakan dalam respons.

- totalRecords— Diperlukan: Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah total catatan dalam tanggapan.

Kesalahan

- UnsupportedOperationException
- ExpiredStreamException
- InvalidParameterException
- MemoryLimitExceededException
- StreamRecordsNotFoundException
- ClientTimeoutException
- PreconditionsFailedException
- ThrottlingException
- ConstraintViolationException
- InvalidArgumentException
- IllegalArgumentException
- TooManyRequestsException

Struktur data aliran:

PropertygraphRecord (struktur)

Struktur catatan grafik properti.

Bidang

- commitTimestampInMillis— Ini Diperlukan: Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Waktu di mana komit untuk transaksi diminta, dalam milidetik dari zaman Unix.
- data— Ini Diperlukan: Sebuah PropertygraphData objek.

  Catatan perubahan Gremlin atau OpenCypher yang diserialisasi.
- eventId— Ini Diperlukan: Ini adalah array peta pasangan kunci-nilai di mana:

  Setiap kunci adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Setiap nilai adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi urutan catatan perubahan aliran.
- isLastOp— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Hanya hadir jika operasi ini adalah yang terakhir dalam transaksinya. Jika ada, itu diatur ke true. Ini berguna untuk memastikan bahwa seluruh transaksi dikonsumsi.
- op— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Operasi yang menciptakan perubahan.

PropertygraphData (struktur)

Catatan perubahan Gremlin atau OpenCypher.

Bidang

- from— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jika ini adalah tepi (tipe =`e`), ID dari simpul atau `from` simpul sumber yang sesuai.
- id— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID dari elemen Gremlin atau OpenCypher.
- key— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Nama properti. Untuk label elemen, ini adalah`label`.

- to— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Jika ini adalah tepi (tipe =e), ID dari simpul atau `to` simpul target yang sesuai.
- type— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis elemen Gremlin atau OpenCypher ini. Harus menjadi salah satu dari:
  - **v1**- Label Vertex untuk Gremlin, atau label node untuk OpenCypher.
  - **vp**- Properti Vertex untuk Gremlin, atau properti node untuk OpenCypher.
  - **e**- Label tepi dan tepi untuk Gremlin, atau jenis hubungan dan hubungan untuk OpenCypher.
  - **ep**- Properti tepi untuk Gremlin, atau properti hubungan untuk OpenCypher.
- value— Ini Diperlukan: Dokumen, tipe: `document` (konten terbuka protokol-agnostik yang diwakili oleh model data seperti JSON).

  Ini adalah objek JSON yang berisi bidang nilai untuk nilai itu sendiri, dan bidang tipe data untuk tipe data JSON dari nilai itu:

  Example

```
"value": {
          "value": "(the new value"),
          "dataType": "(the JSON datatypenew value")
        }
```

Statistik jalur data Neptunus dan ringkasan grafik APIs

Tindakan statistik grafik properti:

- GetPropertygraphStatistics (tindakan)
- ManagePropertygraphStatistics (tindakan)
- DeletePropertygraphStatistics (tindakan)
- GetPropertygraphSummary (tindakan)

Struktur statistik:

- Statistik (struktur)
- StatisticsSummary (struktur)

- DeleteStatisticsValueMap (struktur)
- RefreshStatisticsIdMap (struktur)
- NodeStructure (struktur)
- EdgeStructure (struktur)
- SubjectStructure (struktur)
- PropertygraphSummaryValueMap (struktur)
- PropertygraphSummary (struktur)

GetPropertygraphStatistics (tindakan)

Nama AWS CLI untuk API ini adalah:. `get-propertygraph-statistics`

Mendapat statistik grafik properti (Gremlin dan OpenCypher).

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. GetStatisticsStatus

Permintaan

- Tidak ada parameter Permintaan.

Respons

- payload — Wajib: Sebuah objek Statistik.

  Statistik untuk data grafik properti.
- status— Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode pengembalian HTTP dari permintaan. Jika permintaan berhasil, kodenya adalah 200. Lihat Kode kesalahan umum untuk permintaan statistik DFE untuk daftar kesalahan umum.

Kesalahan

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException

- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ManagePropertygraphStatistics (tindakan)

Nama AWS CLI untuk API ini adalah:. `manage-propertygraph-statistics`

Mengelola pembuatan dan penggunaan statistik grafik properti.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. ManageStatistics

Permintaan

- mode(dalam CLI:`--mode`) — a StatisticsAutoGenerationMode, dari tipe: `string` (string yang dikodekan UTF-8).

  Mode generasi statistik. Salah satu dari: `DISABLE_AUTOCOMPUTE`ENABLE_AUTOCOMPUTE,, atau`REFRESH`, yang terakhir secara manual memicu pembuatan statistik DFE.

Respons

- payload — Sebuah objek RefreshStatisticsIdMap.

  Ini hanya dikembalikan untuk mode refresh.
- status— Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode pengembalian HTTP dari permintaan. Jika permintaan berhasil, kodenya adalah 200.

Kesalahan

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

DeletePropertygraphStatistics (tindakan)

Nama AWS CLI untuk API ini adalah:. `delete-propertygraph-statistics`

Menghapus statistik untuk data Gremlin dan OpenCypher (grafik properti).

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. DeleteStatistics

Permintaan

- Tidak ada parameter Permintaan.

Respons

- payload — Sebuah objek DeleteStatisticsValueMap.

  Muatan penghapusan.

- status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status pembatalan.

- statusCode— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Kode respons HTTP: 200 jika penghapusan berhasil, atau 204 jika tidak ada statistik untuk dihapus.

Kesalahan

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

GetPropertygraphSummary (tindakan)

Nama AWS CLI untuk API ini adalah:. `get-propertygraph-summary`

Mendapat ringkasan grafik untuk grafik properti.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan neptune-db: IAM di cluster itu. GetGraphSummary

Permintaan

- mode(dalam CLI:`--mode`) — a GraphSummaryType, dari tipe: `string` (string yang dikodekan UTF-8).

  Mode dapat mengambil salah satu dari dua nilai: `BASIC` (default), dan`DETAILED`.

Respons

- payload — Sebuah objek PropertygraphSummaryValueMap.

  Payload yang berisi respon ringkasan grafik properti.

- statusCode— Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Kode pengembalian HTTP dari permintaan. Jika permintaan berhasil, kodenya adalah 200.

Kesalahan

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Struktur statistik:

Statistik (struktur)

Berisi informasi statistik. Mesin DFE menggunakan informasi tentang data dalam grafik Neptunus Anda untuk membuat trade-off yang efektif saat merencanakan eksekusi kueri. Informasi ini mengambil bentuk statistik yang mencakup apa yang disebut set karakteristik dan statistik predikat yang dapat memandu perencanaan kueri. Lihat Mengelola statistik untuk DFE Neptunus untuk digunakan.

Bidang

- active— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah pembuatan statistik DFE diaktifkan sama sekali atau tidak.
- autoCompute— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Menunjukkan apakah pembuatan statistik otomatis diaktifkan atau tidak.
- date— Ini adalah SyntheticTimestamp _date_time, dari tipe: `string` (string yang dikodekan UTF-8).

  Waktu UTC di mana statistik DFE baru-baru ini dihasilkan.
- note— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Catatan tentang masalah dalam kasus di mana statistik tidak valid.
- signatureInfoIni adalah sebuah StatisticsSummary objek.

  StatisticsSummary Struktur yang berisi:
  - `signatureCount`- Jumlah total tanda tangan di semua set karakteristik.
  - `instanceCount`- Jumlah total instance yang ditetapkan karakteristik.
  - `predicateCount`- Jumlah total predikat unik.
- statisticsId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Melaporkan ID dari proses pembuatan statistik saat ini. Nilai -1 menunjukkan bahwa tidak ada statistik yang dihasilkan.

StatisticsSummary (struktur)

Informasi tentang set karakteristik yang dihasilkan dalam statistik.

Bidang

- instanceCount— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah total instance set karakteristik.
- predicateCount— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah total predikat unik.
- signatureCount— Ini adalah Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah total tanda tangan di semua set karakteristik.

DeleteStatisticsValueMap (struktur)

Muatan untuk DeleteStatistics.

Bidang

- active— Ini adalah Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Status statistik saat ini.
- statisticsId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  ID dari proses pembuatan statistik yang sedang terjadi saat ini.

RefreshStatisticsIdMap (struktur)

Statistik untuk `REFRESH` mode.

Bidang

- statisticsId— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  ID dari proses pembuatan statistik yang sedang terjadi saat ini.

NodeStructure (struktur)

Struktur simpul.

Bidang

- count— Ini adalah Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Jumlah node yang memiliki struktur khusus ini.
- distinctOutgoingEdgeLabels— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar label tepi keluar yang berbeda hadir dalam struktur khusus ini.
- nodeProperties— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar properti simpul yang ada dalam struktur khusus ini.

EdgeStructure (struktur)

Struktur tepi.

Bidang

- count— Ini adalah Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Jumlah tepi yang memiliki struktur khusus ini.
- edgeProperties— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar properti tepi yang ada dalam struktur khusus ini.

SubjectStructure (struktur)

Struktur subjek.

Bidang

- count— Ini adalah Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Jumlah kemunculan struktur khusus ini.
- predicates— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

Daftar predikat hadir dalam struktur khusus ini.

PropertygraphSummaryValueMap (struktur)

Payload untuk respon ringkasan grafik properti.

Bidang

- graphSummaryIni adalah sebuah PropertygraphSummary objek.

  Ringkasan grafik.
- lastStatisticsComputationTime— Ini adalah SyntheticTimestamp _date_time, dari tipe: `string` (string yang dikodekan UTF-8).

  Stempel waktu, dalam format ISO 8601, saat Neptunus terakhir menghitung statistik.
- version— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Versi respons ringkasan grafik ini.

PropertygraphSummary (struktur)

API ringkasan grafik mengembalikan daftar read-only label node dan edge dan kunci properti, bersama dengan jumlah node, tepi, dan properti. Lihat Respons ringkasan grafik untuk grafik properti (PG).

Bidang

- edgeLabels— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar label tepi yang berbeda dalam grafik.
- edgeProperties— Ini adalah LongValuedMap objek Ini adalah array peta pasangan kunci-nilai di mana:

  Setiap kunci adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Setiap nilai adalah Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Daftar properti tepi yang berbeda dalam grafik, bersama dengan jumlah tepi di mana setiap properti digunakan.

- edgeStructures— Ini adalah Array EdgeStructure objek.

  Bidang ini hanya ada ketika mode yang dimintaDETAILED. Ini berisi daftar struktur tepi.
- nodeLabels— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Daftar label node yang berbeda dalam grafik.
- nodeProperties— Ini adalah LongValuedMap objek Ini adalah array peta pasangan kunci-nilai di mana:

  Setiap kunci adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Setiap nilai adalah Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Jumlah properti node yang berbeda dalam grafik.
- nodeStructures— Ini adalah Array NodeStructure objek.

  Bidang ini hanya ada ketika mode yang dimintaDETAILED. Ini berisi daftar struktur simpul.
- numEdgeLabels— Ini adalah Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Jumlah label tepi yang berbeda dalam grafik.
- numEdgeProperties— Ini adalah Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Jumlah properti tepi yang berbeda dalam grafik.
- numEdges— Ini adalah Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Jumlah tepi dalam grafik.
- numNodeLabels— Ini adalah Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Jumlah label node yang berbeda dalam grafik.
- numNodeProperties— Ini adalah Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Daftar properti node yang berbeda dalam grafik, bersama dengan jumlah node di mana setiap properti digunakan.
- numNodes— Ini adalah Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Jumlah node dalam grafik.
- totalEdgePropertyValues— Ini adalah Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Jumlah total penggunaan semua properti tepi.

- totalNodePropertyValues— Ini adalah Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Jumlah total penggunaan semua properti node.

API pemrosesan data Neptunus Neptunus

Tindakan pemrosesan data:

- Mulai MLData ProcessingJob (tindakan)
- Daftar MLData ProcessingJobs (tindakan)
- Dapatkan MLData ProcessingJob (tindakan)
- Batalkan MLData ProcessingJob (tindakan)

Struktur tujuan umum ML:

- MlResourceDefinition (struktur)
- MlConfigDefinition (struktur)

Mulai MLData ProcessingJob (tindakan)

Nama AWS CLI untuk API ini adalah:. `start-ml-data-processing-job`

Membuat pekerjaan pemrosesan data Neptunus ML baru untuk memproses data grafik yang diekspor dari Neptunus untuk pelatihan. Lihat `dataprocessing`perintahnya.

Saat menjalankan operasi ini di kluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan IAM Neptunus-DB:start di cluster tersebut. MLModel DataProcessingJob

Permintaan

- configFileName(dalam CLI:`--config-file-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  File spesifikasi data yang menjelaskan cara memuat data grafik yang diekspor untuk pelatihan. File secara otomatis dihasilkan oleh kit alat ekspor Neptune. Nilai default-nya `training-data-configuration.json`.

- id(dalam CLI:`--id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengenal unik untuk pekerjaan baru. Defaultnya adalah UUID yang dibuat secara otomatis.

- inputDataS3Location(dalam CLI:`--input-data-s3-location`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  URI lokasi Amazon S3 tempat Anda ingin SageMaker mengunduh data yang diperlukan untuk menjalankan pekerjaan pemrosesan data.

- modelType(dalam CLI:`--model-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Salah satu dari dua jenis model yang didukung Neptunus ML saat ini: model grafik heterogen `heterogeneous` (), dan grafik pengetahuan (). `kge` Defaultnya tidak ada. Jika tidak ditentukan, Neptunus ML memilih jenis model secara otomatis berdasarkan data.

- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama Sumber Daya Amazon (ARN) dari peran IAM yang SageMaker dapat diasumsikan untuk melakukan tugas atas nama Anda. Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

- previousDataProcessingJobId(dalam CLI:`--previous-data-processing-job-id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ID pekerjaan dari pekerjaan pemrosesan data yang telah selesai dijalankan pada versi data yang lebih lama.

- processedDataS3Location(dalam CLI:`--processed-data-s3-location`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  URI lokasi Amazon S3 tempat Anda SageMaker ingin menyimpan hasil pekerjaan pemrosesan data.

- processingInstanceType(dalam CLI:`--processing-instance-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis instance ML yang digunakan selama pemrosesan data. Memorinya harus cukup besar untuk menahan set data yang diproses. Defaultnya adalah tipe ml.r5 terkecil yang memorinya sepuluh kali lebih besar dari ukuran data grafik yang diekspor pada disk.

- processingInstanceVolumeSizeInGB(dalam CLI:`--processing-instance-volume-size-in-gb`) — Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

Ukuran volume disk dari instance pemrosesan. Data input dan data yang diproses disimpan pada disk, sehingga ukuran volume harus cukup besar untuk menahan kedua set data. Default-nya adalah 0. Jika tidak ditentukan atau 0, Neptunus ML memilih ukuran volume secara otomatis berdasarkan ukuran data.

- processingTimeOutInSeconds(dalam CLI:`--processing-time-out-in-seconds`) — Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Batas waktu dalam hitungan detik untuk pekerjaan pemrosesan data. Defaultnya adalah 86.400 (1 hari).

- s3OutputEncryptionKMSKey(dalam CLI:`--s-3-output-encryption-kms-key`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Kunci Amazon Key Management Service (Amazon KMS) yang SageMaker digunakan untuk mengenkripsi output dari pekerjaan pemrosesan. Defaultnya tidak ada.

- sagemakerIamRoleArn(dalam CLI:`--sagemaker-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM untuk eksekusi. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

- securityGroupIds(dalam CLI:`--security-group-ids`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Grup keamanan VPC. IDs Default-nya adalah Tidak Ada.

- subnets(dalam CLI:`--subnets`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Subnet di VPC Neptunus. IDs Default-nya adalah Tidak Ada.

- volumeEncryptionKMSKey(dalam CLI:`--volume-encryption-kms-key`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Kunci Amazon Key Management Service (Amazon KMS) yang SageMaker digunakan untuk mengenkripsi data pada volume penyimpanan yang dilampirkan ke instans komputasi ML yang menjalankan tugas pelatihan. Default-nya adalah Tidak Ada.

Respons

- arn— String, tipe: `string` (string yang dikodekan UTF-8).

ARN dari pekerjaan pemrosesan data.

- creationTimeInMillis— Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Waktu yang dibutuhkan untuk membuat pekerjaan pemrosesan baru, dalam milidetik.

- id— String, tipe: `string` (string yang dikodekan UTF-8).

  ID unik dari pekerjaan pemrosesan data baru.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Daftar MLData ProcessingJobs (tindakan)

Nama AWS CLI untuk API ini adalah:. `list-ml-data-processing-jobs`

Mengembalikan daftar pekerjaan pemrosesan data Neptunus Neptunus. Lihat Membuat daftar pekerjaan pemrosesan data aktif menggunakan perintah pemrosesan data Neptunus ML.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan IAM Neptune-DB:List di cluster tersebut. MLData ProcessingJobs

Permintaan

- maxItems(dalam CLI:`--max-items`) — Daftar ListMLDataProcessingJobsInputMaxItemsInteger, tipe: `integer` (bilangan bulat 32-bit yang ditandatangani), tidak kurang dari 1 atau lebih dari 1024? st? s.

  Jumlah maksimum item yang akan dikembalikan (dari 1 hingga 1024; defaultnya adalah 10).
- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM yang menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

Respons

- ids— String, tipe: `string` (string yang dikodekan UTF-8).

  Halaman yang mencantumkan pekerjaan pemrosesan data IDs.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Dapatkan MLData ProcessingJob (tindakan)

Nama AWS CLI untuk API ini adalah:. `get-ml-data-processing-job`

Mengambil informasi tentang pekerjaan pemrosesan data tertentu. Lihat `dataprocessing`perintahnya.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan Neptune-DB:Neptune-db:get IAM di cluster itu. MLData ProcessingJobStatus

Permintaan

- id(dalam CLI:`--id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik dari pekerjaan pemrosesan data yang akan diambil.
- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM yang menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

Respons

- id— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik dari pekerjaan pemrosesan data ini.
- processingJob — Sebuah objek MlResourceDefinition.

  Definisi pekerjaan pemrosesan data.
- status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status pekerjaan pemrosesan data.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException

- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Batalkan MLData ProcessingJob (tindakan)

Nama AWS CLI untuk API ini adalah:. `cancel-ml-data-processing-job`

Membatalkan pekerjaan pemrosesan data Neptunus Neptunus. Lihat `dataprocessing`perintahnya.

Saat menjalankan operasi ini di kluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan IAM Neptune-DB:Cancel di cluster tersebut. MLData ProcessingJob

Permintaan

- clean(dalam CLI:`--clean`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke`TRUE`, bendera ini menentukan bahwa semua artefak Neptunus ML S3 harus dihapus ketika pekerjaan dihentikan. Nilai default-nya `FALSE`.
- id(dalam CLI:`--id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik dari pekerjaan pemrosesan data.
- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM yang menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

Respons

- status— String, tipe: `string` (string yang dikodekan UTF-8).

Status permintaan pembatalan.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Struktur tujuan umum ML:

MlResourceDefinition (struktur)

Mendefinisikan sumber daya Neptunus Neptunus.

Bidang

- arn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Sumber daya ARN.
- cloudwatchLogUrl— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  URL CloudWatch log untuk sumber daya.
- failureReason— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Alasan kegagalan, jika terjadi kegagalan.
- name— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

Nama sumber daya.

- outputLocation— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Lokasi output.

- status— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Status sumber daya.

MlConfigDefinition (struktur)

Berisi konfigurasi Neptunus Neptunus.

Bidang

- arn— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  ARN untuk konfigurasi.

- name— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama konfigurasi.

API pelatihan model Neptunus Neptunus

Tindakan pelatihan model:

- Mulai MLModel TrainingJob (tindakan)
- Daftar MLModel TrainingJobs (tindakan)
- Dapatkan MLModel TrainingJob (tindakan)
- Batalkan MLModel TrainingJob (tindakan)

Struktur pelatihan model:

- CustomModelTrainingParameters (struktur)

Mulai MLModel TrainingJob (tindakan)

Nama AWS CLI untuk API ini adalah:. `start-ml-model-training-job`

Menciptakan pekerjaan pelatihan model Neptunus ML baru. Lihat Pelatihan model menggunakan `modeltraining` perintah.

Saat menjalankan operasi ini di kluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan IAM Neptunus-DB:start di cluster tersebut. MLModel TrainingJob

Permintaan

- baseProcessingInstanceType(dalam CLI:`--base-processing-instance-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis instance ML yang digunakan dalam mempersiapkan dan mengelola pelatihan model ML. Ini adalah instance CPU yang dipilih berdasarkan persyaratan memori untuk memproses data dan model pelatihan.
- customModelTrainingParameters(dalam CLI:`--custom-model-training-parameters`) — Sebuah CustomModelTrainingParameters objek.

  Konfigurasi untuk pelatihan model khusus. Ini adalah objek JSON.
- dataProcessingJobId(dalam CLI:`--data-processing-job-id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID pekerjaan dari pekerjaan pemrosesan data yang telah selesai yang telah menciptakan data yang akan dikerjakan oleh pelatihan.
- enableManagedSpotTraining(dalam CLI:`--enable-managed-spot-training`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Mengoptimalkan biaya pelatihan model pembelajaran mesin dengan menggunakan instans spot Amazon Elastic Compute Cloud. Nilai default-nya `False`.
- id(dalam CLI:`--id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengenal unik untuk pekerjaan baru. Defaultnya adalah UUID yang dibuat secara otomatis.
- maxHPONumberOfTrainingJobs(dalam CLI:`--max-hpo-number-of-training-jobs`) — Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

Jumlah total maksimum pekerjaan pelatihan untuk memulai untuk pekerjaan tuning hyperparameter. Defaultnya adalah 2. Neptunus ML secara otomatis menyetel hiperparameter model pembelajaran mesin. Untuk mendapatkan model yang berkinerja baik, gunakan setidaknya 10 pekerjaan (dengan kata lain, atur `maxHPONumberOfTrainingJobs` ke 10). Secara umum, semakin banyak tuning berjalan, semakin baik hasilnya.

- maxHPOParallelTrainingJobs(dalam CLI:`--max-hpo-parallel-training-jobs`) — Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah maksimum pekerjaan pelatihan paralel untuk memulai untuk pekerjaan tuning hyperparameter. Defaultnya adalah 2. Jumlah pekerjaan paralel yang dapat Anda jalankan dibatasi oleh sumber daya yang tersedia pada instans pelatihan Anda.

- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM yang menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

- previousModelTrainingJobId(dalam CLI:`--previous-model-training-job-id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ID pekerjaan dari pekerjaan pelatihan model yang telah selesai yang ingin Anda perbarui secara bertahap berdasarkan data yang diperbarui.

- s3OutputEncryptionKMSKey(dalam CLI:`--s-3-output-encryption-kms-key`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Kunci Amazon Key Management Service (KMS) yang SageMaker digunakan untuk mengenkripsi output dari pekerjaan pemrosesan. Defaultnya tidak ada.

- sagemakerIamRoleArn(dalam CLI:`--sagemaker-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM untuk SageMaker eksekusi. Ini harus terdaftar di grup parameter cluster DB Anda atau kesalahan akan terjadi.

- securityGroupIds(dalam CLI:`--security-group-ids`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Grup keamanan VPC. IDs Default-nya adalah Tidak Ada.

- subnets(dalam CLI:`--subnets`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Subnet di VPC Neptunus. IDs Default-nya adalah Tidak Ada.

- trainingInstanceType(dalam CLI:`--training-instance-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis instance ML yang digunakan untuk pelatihan model. Semua model Neptune ML mendukung pelatihan CPU, GPU, dan MultiGPU. Nilai default-nya `ml.p3.2xlarge`. Memilih jenis instans yang tepat untuk pelatihan tergantung pada jenis tugas, ukuran grafik, dan anggaran Anda.

- trainingInstanceVolumeSizeInGB(dalam CLI:`--training-instance-volume-size-in-gb`) — Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Ukuran volume disk dari instance pelatihan. Input data dan model output disimpan dalam disk, sehingga ukuran volume harus cukup besar untuk menahan kedua set data. Default-nya adalah 0. Jika tidak ditentukan atau 0, Neptunus ML memilih ukuran volume disk berdasarkan rekomendasi yang dihasilkan dalam langkah pemrosesan data.

- trainingTimeOutInSeconds(dalam CLI:`--training-time-out-in-seconds`) — Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Batas waktu dalam hitungan detik untuk pekerjaan pelatihan. Defaultnya adalah 86.400 (1 hari).

- trainModelS3Location(dalam CLI:`--train-model-s3-location`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Lokasi di Amazon S3 tempat artefak model akan disimpan.

- volumeEncryptionKMSKey(dalam CLI:`--volume-encryption-kms-key`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Kunci Amazon Key Management Service (KMS) yang SageMaker digunakan untuk mengenkripsi data pada volume penyimpanan yang dilampirkan ke instans komputasi ML yang menjalankan tugas pelatihan. Default-nya adalah Tidak Ada.

Respons

- arn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari pekerjaan pelatihan model baru.

- creationTimeInMillis— Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

Model pelatihan waktu penciptaan lapangan kerja, dalam milidetik.

- id— String, tipe: `string` (string yang dikodekan UTF-8).

  ID unik dari pekerjaan pelatihan model baru.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Daftar MLModel TrainingJobs (tindakan)

Nama AWS CLI untuk API ini adalah:. `list-ml-model-training-jobs`

Daftar pekerjaan pelatihan model Neptunus ML. Lihat Pelatihan model menggunakan `modeltraining` perintah.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan IAM Neptune-DB:Neptune-db:list di cluster tersebut. MLModel TrainingJobs

Permintaan

- maxItems(dalam CLI:`--max-items`) — Daftar MLModelTrainingJobsInputMaxItemsInteger, tipe: `integer` (bilangan bulat 32-bit yang ditandatangani), tidak kurang dari 1 atau lebih dari 1024? st? s.

Jumlah maksimum item yang akan dikembalikan (dari 1 hingga 1024; defaultnya adalah 10).

- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM yang menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

Respons

- ids— String, tipe: `string` (string yang dikodekan UTF-8).

  Halaman daftar pekerjaan pelatihan model IDs.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Dapatkan MLModel TrainingJob (tindakan)

Nama AWS CLI untuk API ini adalah:. `get-ml-model-training-job`

Mengambil informasi tentang pekerjaan pelatihan model Neptunus ML. Lihat Pelatihan model menggunakan `modeltraining` perintah.

Saat menjalankan operasi ini di kluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan Neptunus-DB:get IAM di cluster tersebut. MLModel TrainingJobStatus

Permintaan

- id(dalam CLI:`--id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik dari pekerjaan pelatihan model untuk diambil.

- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM yang menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

Respons

- hpoJob — Sebuah objek MlResourceDefinition.

  Pekerjaan HPO.

- id— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik dari pekerjaan pelatihan model ini.

- mlModels – Susunan objek MlConfigDefinition.

  Daftar konfigurasi model ML yang digunakan.

- modelTransformJob — Sebuah objek MlResourceDefinition.

  Model mengubah pekerjaan.

- processingJob — Sebuah objek MlResourceDefinition.

  Pekerjaan pemrosesan data.

- status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status pekerjaan pelatihan model.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Batalkan MLModel TrainingJob (tindakan)

Nama AWS CLI untuk API ini adalah:. `cancel-ml-model-training-job`

Membatalkan pekerjaan pelatihan model Neptunus Neptunus. Lihat Pelatihan model menggunakan `modeltraining` perintah.

Saat menjalankan operasi ini di kluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan IAM Neptune-DB:Cancel di cluster tersebut. MLModel TrainingJob

Permintaan

- clean(dalam CLI:`--clean`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika disetel ke`TRUE`, flag ini menetapkan bahwa semua artefak Amazon S3 harus dihapus ketika pekerjaan dihentikan. Nilai default-nya `FALSE`.
- id(dalam CLI:`--id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik dari pekerjaan pelatihan model yang akan dibatalkan.
- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

ARN dari peran IAM yang menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

Respons

- status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status pembatalan.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Struktur pelatihan model:

CustomModelTrainingParameters (struktur)

Berisi parameter pelatihan model khusus. Lihat Model khusus di Neptunus ML.

Bidang

- sourceS3DirectoryPath— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

Jalur ke lokasi Amazon S3 tempat modul Python yang mengimplementasikan model Anda berada. Ini harus menunjuk ke lokasi Amazon S3 yang valid yang berisi, setidaknya, skrip pelatihan, skrip transformasi, dan file. `model-hpo-configuration.json`

- trainingEntryPointScript— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama titik masuk dalam modul skrip Anda yang melakukan pelatihan model dan menggunakan hyperparameters sebagai argumen baris perintah, termasuk hyperparameters tetap. Nilai default-nya `training.py`.

- transformEntryPointScript— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama titik masuk dalam modul skrip Anda yang harus dijalankan setelah model terbaik dari pencarian hyperparameter telah diidentifikasi, untuk menghitung artefak model yang diperlukan untuk penerapan model. Itu harus dapat berjalan tanpa argumen baris perintah. Defaultnya adalah. `transform.py`

Model Neptunus Neptunus mengubah API

Tindakan transformasi model:

- Mulai MLModel TransformJob (tindakan)
- Daftar MLModel TransformJobs (tindakan)
- Dapatkan MLModel TransformJob (tindakan)
- Batalkan MLModel TransformJob (tindakan)

Struktur transformasi model:

- CustomModelTransformParameters (struktur)

Mulai MLModel TransformJob (tindakan)

Nama AWS CLI untuk API ini adalah:. `start-ml-model-transform-job`

Menciptakan pekerjaan transformasi model baru. Lihat Menggunakan model terlatih untuk menghasilkan artefak model baru.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan IAM Neptune-DB:start di cluster tersebut. MLModel TransformJob

Permintaan

- baseProcessingInstanceType(dalam CLI:`--base-processing-instance-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis instance ML yang digunakan dalam mempersiapkan dan mengelola pelatihan model ML. Ini adalah instance komputasi ML yang dipilih berdasarkan persyaratan memori untuk memproses data dan model pelatihan.
- baseProcessingInstanceVolumeSizeInGB(dalam CLI:`--base-processing-instance-volume-size-in-gb`) — Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Ukuran volume disk dari instance pelatihan dalam gigabyte. Default-nya adalah 0. Input data dan model output disimpan dalam disk, sehingga ukuran volume harus cukup besar untuk menahan kedua set data. Jika tidak ditentukan atau 0, Neptunus ML memilih ukuran volume disk berdasarkan rekomendasi yang dihasilkan dalam langkah pemrosesan data.
- customModelTransformParameters(dalam CLI:`--custom-model-transform-parameters`) — Sebuah CustomModelTransformParameters objek.

  Informasi konfigurasi untuk transformasi model menggunakan model kustom. `customModelTransformParameters`Objek berisi bidang berikut, yang harus memiliki nilai yang kompatibel dengan parameter model yang disimpan dari pekerjaan pelatihan:
- dataProcessingJobId(dalam CLI:`--data-processing-job-id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ID pekerjaan dari pekerjaan pemrosesan data yang telah selesai. Anda harus menyertakan salah satu `dataProcessingJobId` dan`mlModelTrainingJobId`, atau `atrainingJobName`.
- id(dalam CLI:`--id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengenal unik untuk pekerjaan baru. Defaultnya adalah UUID yang dibuat secara otomatis.
- mlModelTrainingJobId(dalam CLI:`--ml-model-training-job-id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ID pekerjaan dari pekerjaan pelatihan model yang telah selesai. Anda harus menyertakan salah satu `dataProcessingJobId` dan`mlModelTrainingJobId`, atau `atrainingJobName`.

- modelTransformOutputS3Location(dalam CLI:`--model-transform-output-s3-location`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Lokasi di Amazon S3 tempat artefak model akan disimpan.

- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM yang menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

- s3OutputEncryptionKMSKey(dalam CLI:`--s-3-output-encryption-kms-key`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Kunci Amazon Key Management Service (KMS) yang SageMaker digunakan untuk mengenkripsi output dari pekerjaan pemrosesan. Defaultnya tidak ada.

- sagemakerIamRoleArn(dalam CLI:`--sagemaker-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM untuk eksekusi. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

- securityGroupIds(dalam CLI:`--security-group-ids`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Grup keamanan VPC. IDs Default-nya adalah Tidak Ada.

- subnets(dalam CLI:`--subnets`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Subnet di VPC Neptunus. IDs Default-nya adalah Tidak Ada.

- trainingJobName(dalam CLI:`--training-job-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Nama pekerjaan SageMaker pelatihan yang telah selesai. Anda harus menyertakan salah satu `dataProcessingJobId` dan`mlModelTrainingJobId`, atau `atrainingJobName`.

- volumeEncryptionKMSKey(dalam CLI:`--volume-encryption-kms-key`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Kunci Amazon Key Management Service (KMS) yang SageMaker digunakan untuk mengenkripsi data pada volume penyimpanan yang dilampirkan ke instans komputasi ML yang menjalankan tugas pelatihan. Default-nya adalah Tidak Ada.

Respons

- arn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari model mengubah pekerjaan.
- creationTimeInMillis— Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Waktu pembuatan model mengubah pekerjaan, dalam milidetik.
- id— String, tipe: `string` (string yang dikodekan UTF-8).

  ID unik dari pekerjaan transformasi model baru.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Daftar MLModel TransformJobs (tindakan)

Nama AWS CLI untuk API ini adalah:. `list-ml-model-transform-jobs`

Mengembalikan daftar pekerjaan model transformasi IDs. Lihat Menggunakan model terlatih untuk menghasilkan artefak model baru.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan IAM Neptune-DB:List di cluster tersebut. MLModel TransformJobs

Permintaan

- maxItems(dalam CLI:`--max-items`) — Daftar ListMLModelTransformJobsInputMaxItemsInteger, tipe: `integer` (bilangan bulat 32-bit yang ditandatangani), tidak kurang dari 1 atau lebih dari 1024? st? s.

  Jumlah maksimum item yang akan dikembalikan (dari 1 hingga 1024; defaultnya adalah 10).
- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM yang menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

Respons

- ids— String, tipe: `string` (string yang dikodekan UTF-8).

  Halaman dari daftar transformasi model IDs.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Dapatkan MLModel TransformJob (tindakan)

Nama AWS CLI untuk API ini adalah:. `get-ml-model-transform-job`

Mendapat informasi tentang pekerjaan transformasi model tertentu. Lihat Menggunakan model terlatih untuk menghasilkan artefak model baru.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan Neptunus-DB:get IAM di cluster tersebut. MLModel TransformJobStatus

Permintaan

- id(dalam CLI:`--id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik dari pekerjaan transformasi model yang akan direetrieved.
- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM yang menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

Respons

- baseProcessingJob — Sebuah objek MlResourceDefinition.

  Pekerjaan pemrosesan data dasar.
- id— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik dari pekerjaan transformasi model yang akan diambil.
- models – Susunan objek MlConfigDefinition.

  Daftar informasi konfigurasi untuk model yang digunakan.
- remoteModelTransformJob — Sebuah objek MlResourceDefinition.

  Pekerjaan transformasi model jarak jauh.
- status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status pekerjaan transformasi model.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Batalkan MLModel TransformJob (tindakan)

Nama AWS CLI untuk API ini adalah:. `cancel-ml-model-transform-job`

Membatalkan pekerjaan transformasi model tertentu. Lihat Menggunakan model terlatih untuk menghasilkan artefak model baru.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan IAM Neptune-DB:Cancel di cluster tersebut. MLModel TransformJob

Permintaan

- clean(dalam CLI:`--clean`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

  Jika bendera ini disetel ke`TRUE`, semua artefak Neptunus ML S3 harus dihapus saat pekerjaan dihentikan. Nilai default-nya `FALSE`.
- id(dalam CLI:`--id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID unik dari pekerjaan transformasi model yang akan dibatalkan.
- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

ARN dari peran IAM yang menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

Respons

- status— String, tipe: `string` (string yang dikodekan UTF-8).

  status pembatalan.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Struktur transformasi model:

CustomModelTransformParameters (struktur)

Berisi parameter transformasi model kustom. Lihat Menggunakan model terlatih untuk menghasilkan artefak model baru.

Bidang

- sourceS3DirectoryPath— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

Jalur ke lokasi Amazon S3 tempat modul Python yang mengimplementasikan model Anda berada. Ini harus menunjuk ke lokasi Amazon S3 yang valid yang berisi, setidaknya, skrip pelatihan, skrip transformasi, dan file. `model-hpo-configuration.json`

- transformEntryPointScript— Ini adalah String, tipe: `string` (string yang dikodekan UTF-8).

  Nama titik masuk dalam modul skrip Anda yang harus dijalankan setelah model terbaik dari pencarian hyperparameter telah diidentifikasi, untuk menghitung artefak model yang diperlukan untuk penerapan model. Itu harus dapat berjalan tanpa argumen baris perintah. Nilai default-nya `transform.py`.

API titik akhir inferensi Neptunus ML

Tindakan titik akhir inferensi:

- Buat MLEndpoint (tindakan)
- Daftar MLEndpoints (tindakan)
- Dapatkan MLEndpoint (tindakan)
- Hapus MLEndpoint (tindakan)

Buat MLEndpoint (tindakan)

Nama AWS CLI untuk API ini adalah:. `create-ml-endpoint`

Membuat titik akhir inferensi Neptunus ML baru yang memungkinkan Anda menanyakan satu model spesifik yang dibuat oleh proses pelatihan model. Lihat Mengelola titik akhir inferensi menggunakan perintah endpoints.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan Neptune-DB:Create IAM di cluster tersebut. MLEndpoint

Permintaan

- id(dalam CLI:`--id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik untuk titik akhir inferensi baru. Defaultnya adalah nama stempel waktu yang dibuat secara otomatis.

- instanceCount(dalam CLI:`--instance-count`) — Integer, tipe: `integer` (integer 32-bit yang ditandatangani).

  Jumlah minimum EC2 instans Amazon untuk diterapkan ke titik akhir untuk prediksi. Defaultnya adalah 1

- instanceType(dalam CLI:`--instance-type`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis instance Neptunus ML yang digunakan untuk servis online. Nilai default-nya `ml.m5.xlarge`. Memilih instans ML untuk titik akhir inferensi bergantung pada jenis tugas, ukuran grafik, dan anggaran Anda.

- mlModelTrainingJobId(dalam CLI:`--ml-model-training-job-id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Id pekerjaan dari pekerjaan pelatihan model yang telah diselesaikan yang telah menciptakan model yang akan ditunjukkan oleh titik akhir inferensi. Anda harus menyediakan salah satu `mlModelTrainingJobId` ataumlModelTransformJobId.

- mlModelTransformJobId(dalam CLI:`--ml-model-transform-job-id`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Id pekerjaan dari pekerjaan transformasi model yang telah selesai. Anda harus menyediakan salah satu `mlModelTrainingJobId` ataumlModelTransformJobId.

- modelName(dalam CLI:`--model-name`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Jenis model untuk pelatihan. Secara default model Neptunus Neptunus secara otomatis didasarkan pada `modelType` yang digunakan dalam pemrosesan data, tetapi Anda dapat menentukan jenis model yang berbeda di sini. Defaultnya adalah `rgcn` untuk grafik heterogen dan `kge` untuk grafik pengetahuan. Satu-satunya nilai yang valid untuk grafik heterogen adalah. `rgcn` Nilai yang valid untuk grafik pengetahuan adalah:`kge`,, `transedistmult`, dan`rotate`.

- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus terdaftar di grup parameter cluster DB Anda atau kesalahan akan dilemparkan.

- update(dalam CLI:`--update`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

Jika disetel ke`true`, `update` menunjukkan bahwa ini adalah permintaan pembaruan. Nilai default-nya `false`. Anda harus menyediakan salah satu `mlModelTrainingJobId` ataum`lModelTransformJobId`.

- volumeEncryptionKMSKey(dalam CLI:`--volume-encryption-kms-key`) — String, tipe: `string` (string yang dikodekan UTF-8).

  Kunci Amazon Key Management Service (Amazon KMS) yang SageMaker digunakan untuk mengenkripsi data pada volume penyimpanan yang dilampirkan ke instans komputasi ML yang menjalankan tugas pelatihan. Default-nya adalah Tidak Ada.

Respons

- arn— String, tipe: `string` (string yang dikodekan UTF-8).

  ARN untuk titik akhir inferensi baru.
- creationTimeInMillis— Panjang, tipe: `long` (integer 64-bit yang ditandatangani).

  Waktu pembuatan titik akhir, dalam milidetik.
- id— String, tipe: `string` (string yang dikodekan UTF-8).

  ID unik dari titik akhir inferensi baru.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException

- TooManyRequestsException

Daftar MLEndpoints (tindakan)

Nama AWS CLI untuk API ini adalah:. `list-ml-endpoints`

Daftar titik akhir inferensi yang ada. Lihat Mengelola titik akhir inferensi menggunakan perintah endpoints.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan IAM Neptune-DB:List di cluster tersebut. MLEndpoints

Permintaan

- maxItems(dalam CLI:`--max-items`) — Daftar MLEndpointsInputMaxItemsInteger, tipe: `integer` (bilangan bulat 32-bit yang ditandatangani), tidak kurang dari 1 atau lebih dari 1024? st? s.

  Jumlah maksimum item yang akan dikembalikan (dari 1 hingga 1024; defaultnya adalah 10.
- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM yang menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

Respons

- ids— String, tipe: `string` (string yang dikodekan UTF-8).

  Halaman dari daftar titik akhir IDs inferensi.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException

- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Dapatkan MLEndpoint (tindakan)

Nama AWS CLI untuk API ini adalah:. `get-ml-endpoint`

Mengambil detail tentang titik akhir inferensi. Lihat Mengelola titik akhir inferensi menggunakan perintah endpoints.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan IAM Neptune-DB:Get Status di cluster tersebut MLEndpoint.

Permintaan

- id(dalam CLI:`--id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik dari titik akhir inferensi.
- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM yang menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus tercantum dalam grup parameter cluster DB Anda atau kesalahan akan terjadi.

Respons

- endpoint — Sebuah objek MlResourceDefinition.

  Definisi titik akhir.
- endpointConfig — Sebuah objek MlConfigDefinition.

Konfigurasi titik akhir

- id— String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik dari titik akhir inferensi.

- status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status titik akhir inferensi.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Hapus MLEndpoint (tindakan)

Nama AWS CLI untuk API ini adalah:. `delete-ml-endpoint`

Membatalkan pembuatan titik akhir inferensi Neptunus ML. Lihat Mengelola titik akhir inferensi menggunakan perintah endpoints.

Saat menjalankan operasi ini di cluster Neptunus yang mengaktifkan autentikasi IAM, pengguna IAM atau peran yang membuat permintaan harus memiliki kebijakan yang dilampirkan yang memungkinkan tindakan IAM Neptune-DB:delete di cluster tersebut. MLEndpoint

Permintaan

- clean(dalam CLI:`--clean`) — Boolean, dari tipe: `boolean` (nilai Boolean (benar atau salah)).

Jika bendera ini disetel keTRUE, semua artefak Neptunus ML S3 harus dihapus saat pekerjaan dihentikan. Nilai default-nya FALSE.

- id(dalam CLI:`--id`) - Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pengidentifikasi unik dari titik akhir inferensi.

- neptuneIamRoleArn(dalam CLI:`--neptune-iam-role-arn`) — String, tipe: `string` (string yang dikodekan UTF-8).

  ARN dari peran IAM menyediakan akses Neptunus ke dan sumber daya Amazon S3. SageMaker Ini harus terdaftar di grup parameter cluster DB Anda atau kesalahan akan dilemparkan.

Respons

- status— String, tipe: `string` (string yang dikodekan UTF-8).

  Status pembatalan.

Kesalahan

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Pengecualian API jalur data Neptunus

Pengecualian:

- AccessDeniedException (struktur)
- BadRequestException (struktur)
- BulkLoadIdNotFoundException (struktur)
- CancelledByUserException (struktur)
- ClientTimeoutException (struktur)
- ConcurrentModificationException (struktur)
- ConstraintViolationException (struktur)
- ExpiredStreamException (struktur)
- FailureByQueryException (struktur)
- IllegalArgumentException (struktur)
- InternalFailureException (struktur)
- InvalidArgumentException (struktur)
- InvalidNumericDataException (struktur)
- InvalidParameterException (struktur)
- LoadUrlAccessDeniedException (struktur)
- MalformedQueryException (struktur)
- MemoryLimitExceededException (struktur)
- MethodNotAllowedException (struktur)
- MissingParameterException (struktur)
- MLResourceNotFoundException (struktur)
- ParsingException (struktur)
- PreconditionsFailedException (struktur)
- QueryLimitExceededException (struktur)
- QueryLimitException (struktur)
- QueryTooLargeException (struktur)
- ReadOnlyViolationException (struktur)
- S3Exception (struktur)
- ServerShutdownException (struktur)
- StatisticsNotAvailableException (struktur)
- StreamRecordsNotFoundException (struktur)

- ThrottlingException (struktur)
- TimeLimitExceededException (struktur)
- TooManyRequestsException (struktur)
- UnsupportedOperationException (struktur)
- UnloadUrlAccessDeniedException (struktur)

AccessDeniedException (struktur)

Dibesarkan jika terjadi kegagalan otentikasi atau otorisasi.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

BadRequestException (struktur)

Dibesarkan ketika permintaan diajukan yang tidak dapat diproses.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang buruk.

BulkLoadIdNotFoundException (struktur)

Diangkat ketika ID pekerjaan beban massal yang ditentukan tidak dapat ditemukan.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID pekerjaan beban massal yang tidak dapat ditemukan.

CancelledByUserException (struktur)

Dibesarkan saat pengguna membatalkan permintaan.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

ClientTimeoutException (struktur)

Dibesarkan ketika permintaan habis waktu di klien.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

Kode status HTTP dikembalikan dengan pengecualian.

- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

ConcurrentModificationException (struktur)

Diangkat ketika permintaan mencoba untuk memodifikasi data yang secara bersamaan sedang dimodifikasi oleh proses lain.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

ConstraintViolationException (struktur)

Dibesarkan ketika nilai di bidang permintaan tidak memenuhi batasan yang diperlukan.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

ID permintaan yang dimaksud.

ExpiredStreamException (struktur)

Diangkat saat permintaan mencoba mengakses aliran yang telah kedaluwarsa.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

FailureByQueryException (struktur)

Dibesarkan saat permintaan gagal.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

IllegalArgumentException (struktur)

Diangkat ketika argumen dalam permintaan tidak didukung.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

InternalFailureException (struktur)

Dibesarkan ketika pemrosesan permintaan gagal secara tak terduga.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

InvalidArgumentException (struktur)

Diangkat ketika argumen dalam permintaan memiliki nilai yang tidak valid.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.

- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

InvalidNumericDataException (struktur)

Diangkat ketika data numerik yang tidak valid ditemui saat melayani permintaan.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

InvalidParameterException (struktur)

Dibesarkan ketika nilai parameter tidak valid.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang menyertakan parameter yang tidak valid.

LoadUrlAccessDeniedException (struktur)

Dibesarkan saat akses ditolak ke URL pemuatan tertentu.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

MalformedQueryException (struktur)

Diangkat ketika kueri dikirimkan yang secara sintaksis salah atau tidak lulus validasi tambahan.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan kueri yang salah format.

MemoryLimitExceededException (struktur)

Dibesarkan ketika permintaan gagal karena sumber daya memori yang tidak mencukupi. Permintaan dapat dicoba lagi.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

Pesan terperinci yang menjelaskan masalah.

- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang gagal.

MethodNotAllowedException (struktur)

Dibesarkan ketika metode HTTP yang digunakan oleh permintaan tidak didukung oleh endpoint yang digunakan.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.

- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.

- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

MissingParameterException (struktur)

Dibesarkan ketika parameter yang diperlukan hilang.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.

- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.

- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan di mana parameter hilang.

MLResourceNotFoundException (struktur)

Dibesarkan ketika sumber daya pembelajaran mesin tertentu tidak dapat ditemukan.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

ParsingException (struktur)

Dibesarkan saat masalah parsing ditemui.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

PreconditionsFailedException (struktur)

Dibesarkan ketika prasyarat untuk memproses permintaan tidak terpenuhi.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

Kode status HTTP dikembalikan dengan pengecualian.

- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

QueryLimitExceededException (struktur)

Dibesarkan ketika jumlah kueri aktif melebihi apa yang dapat diproses server. Kueri yang dimaksud dapat dicoba ulang ketika sistem kurang sibuk.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang melebihi batas.

QueryLimitException (struktur)

Dibesarkan ketika ukuran kueri melebihi batas sistem.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

ID permintaan yang melebihi batas.

QueryTooLargeException (struktur)

Dibesarkan ketika badan kueri terlalu besar.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang terlalu besar.

ReadOnlyViolationException (struktur)

Diangkat saat permintaan mencoba menulis ke sumber daya hanya-baca.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan di mana parameter hilang.

S3Exception (struktur)

Dibesarkan ketika ada masalah mengakses Amazon S3.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

ServerShutdownException (struktur)

Dibesarkan saat server dimatikan saat memproses permintaan.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

StatisticsNotAvailableException (struktur)

Dibesarkan ketika statistik yang diperlukan untuk memenuhi permintaan tidak tersedia.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.

- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

StreamRecordsNotFoundException (struktur)

Diangkat saat rekaman aliran yang diminta oleh kueri tidak dapat ditemukan.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

ThrottlingException (struktur)

Dinaikkan ketika tingkat permintaan melebihi throughput maksimum. Permintaan dapat dicoba lagi setelah menghadapi pengecualian ini.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang tidak dapat diproses karena alasan ini.

TimeLimitExceededException (struktur)

Dibesarkan ketika operasi melebihi batas waktu yang diizinkan untuk itu.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang tidak dapat diproses karena alasan ini.

TooManyRequestsException (struktur)

Dibesarkan ketika jumlah permintaan yang diproses melebihi batas.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang tidak dapat diproses karena alasan ini.

UnsupportedOperationException (struktur)

Diangkat ketika permintaan mencoba untuk memulai operasi yang tidak didukung.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

Kode status HTTP dikembalikan dengan pengecualian.

- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

UnloadUrlAccessDeniedException (struktur)

Diangkat saat akses ditolak ke URL yang merupakan target pembongkaran.

Bidang

- code— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Kode status HTTP dikembalikan dengan pengecualian.
- detailedMessage— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  Pesan terperinci yang menjelaskan masalah.
- requestId— Ini Diperlukan: String, tipe: `string` (string yang dikodekan UTF-8).

  ID permintaan yang dimaksud.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.