Dekomposisi basis data pada AWS

AWS Bimbingan Preskriptif

AWS Bimbingan Preskriptif: Dekomposisi basis data pada AWS

Table of Contents

Dekomposisi basis data pada AWS

Philippe Wanner dan Saurabh Sharma, Amazon Web Services

Oktober 2025 (sejarah dokumen)

Modernisasi database, khususnya dekomposisi database monolitik, adalah alur kerja penting bagi organisasi yang ingin meningkatkan kelincahan, skalabilitas, dan kinerja dalam sistem manajemen data mereka. Ketika bisnis tumbuh dan kebutuhan data mereka menjadi lebih kompleks, database monolitik tradisional sering berjuang untuk mengimbangi. Hal ini menyebabkan kemacetan kinerja, tantangan pemeliharaan, dan kesulitan beradaptasi dengan perubahan persyaratan bisnis.

Berikut ini adalah tantangan umum dengan database monolitik:

- Ketidaksejajaran domain bisnis — Database monolitik sering gagal menyelaraskan teknologi dengan domain bisnis yang berbeda, yang dapat membatasi pertumbuhan organisasi.
- Kendala skalabilitas — Sistem sering mencapai batas penskalaan, yang menciptakan hambatan untuk ekspansi bisnis.
- Kekakuan arsitektur — Struktur yang digabungkan dengan erat membuat sulit untuk memperbarui komponen tertentu tanpa mempengaruhi keseluruhan sistem.
- Degradasi kinerja — Menumbuhkan beban data dan meningkatkan konkurensi pengguna sering menyebabkan memburuknya kinerja sistem.

Berikut ini adalah manfaat dekomposisi database:

- Peningkatan kelincahan bisnis — Dekomposisi memungkinkan adaptasi cepat terhadap perubahan kebutuhan bisnis dan mendukung penskalaan independen.
- Kinerja yang dioptimalkan — Dekomposisi membantu Anda membuat solusi database khusus yang disesuaikan dengan kasus penggunaan tertentu dan secara independen menskalakan setiap database.
- Peningkatan manajemen biaya — Dekomposisi memungkinkan pemanfaatan sumber daya yang lebih efisien dan mengurangi biaya operasional.
- Opsi lisensi fleksibel — Dekomposisi menciptakan peluang untuk beralih dari lisensi kepemilikan yang mahal ke alternatif open source.
- Pemberdayaan inovasi — Dekomposisi memfasilitasi adopsi database yang dibuat khusus untuk beban kerja tertentu.

Audiens yang dituju

Panduan ini membantu arsitek basis data, arsitek solusi cloud, tim pengembangan aplikasi, dan arsitek perusahaan. Ini dirancang untuk membantu Anda menguraikan basis data monolitik menjadi penyimpanan data yang selaras dengan layanan mikro, menerapkan arsitektur basis data berbasis domain, merencanakan strategi migrasi basis data, dan skala operasi basis data untuk memenuhi permintaan bisnis yang terus meningkat. Untuk memahami konsep dan rekomendasi dalam panduan ini, Anda harus terbiasa dengan prinsip-prinsip database relasional dan NoSQL, layanan database terkelola AWS , dan pola arsitektur microservices. Panduan ini dimaksudkan untuk membantu organisasi yang berada dalam tahap awal proyek dekomposisi database.

Tujuan

Panduan ini dapat membantu organisasi Anda mencapai tujuan berikut:

- Kumpulkan persyaratan untuk menguraikan arsitektur target Anda.
- Mengembangkan metodologi sistematis untuk mengevaluasi risiko dan berkomunikasi.
- Buat rencana dekomposisi.
- Tentukan metrik keberhasilan, indikator kinerja utama (KPIs), strategi mitigasi, dan rencana kelangsungan bisnis.
- Menetapkan elastisitas beban kerja yang lebih baik yang membantu Anda mengikuti permintaan bisnis.
- Pelajari cara mengadopsi basis data khusus untuk kasus penggunaan tertentu, yang memungkinkan inovasi.
- Perkuat keamanan dan tata kelola data organisasi Anda.
- Mengurangi biaya melalui hal-hal berikut:
  - Mengurangi biaya lisensi
  - Mengurangi penguncian vendor
  - Peningkatan akses ke dukungan dan inovasi komunitas yang lebih luas
  - Kemampuan untuk memilih teknologi database yang berbeda untuk komponen yang berbeda
  - Migrasi bertahap, yang mengurangi risiko dan menyebarkan biaya dari waktu ke waktu
  - Peningkatan pemanfaatan sumber daya

Tantangan umum dan mengelola tanggung jawab untuk dekomposisi basis data

Dekomposisi basis data adalah proses kompleks yang membutuhkan perencanaan, pelaksanaan, dan manajemen yang cermat. Ketika organisasi berusaha untuk memodernisasi infrastruktur data mereka, mereka sering menghadapi segudang tantangan yang dapat berdampak pada keberhasilan proyek mereka. Bagian ini menjelaskan rintangan umum dan memperkenalkan pendekatan terstruktur untuk mengatasi hambatan ini.

Tantangan umum

Proyek dekomposisi basis data menghadapi beberapa tantangan di seluruh dimensi teknis, orang, dan bisnis. Di bidang teknis, memastikan konsistensi data di seluruh sistem terdistribusi menimbulkan rintangan yang signifikan. Ini juga dapat memiliki dampak kinerja dan stabilitas potensial selama periode transisi, dan Anda harus berintegrasi dengan mulus dengan sistem yang ada. Tantangan terkait orang termasuk kurva belajar yang terkait dengan sistem baru, potensi resistensi terhadap perubahan dari karyawan, dan ketersediaan sumber daya yang diperlukan. Dari perspektif bisnis, proyek harus bersaing dengan risiko pembengkakan waktu, kendala anggaran, dan potensi gangguan bisnis selama proses migrasi.

Mendefinisikan peran dan tanggung jawab

Mengingat tantangan kompleks yang mencakup dimensi teknis, orang, dan bisnis, menetapkan peran dan tanggung jawab yang jelas menjadi penting untuk keberhasilan proyek. Matriks Responsible, Accountable, Consulted, and Informed (RACI) menyediakan struktur yang diperlukan untuk menavigasi tantangan ini. Ini secara eksplisit mendefinisikan siapa yang membuat keputusan, siapa yang melakukan pekerjaan, siapa yang memberikan masukan, dan siapa yang perlu tetap mendapat informasi pada setiap tahap dekomposisi. Kejelasan ini membantu mencegah penundaan yang disebabkan oleh pengambilan keputusan yang ambigu, mendorong keterlibatan pemangku kepentingan yang tepat, dan menciptakan akuntabilitas untuk hasil utama. Tanpa kerangka kerja seperti itu, tim dapat berjuang dengan tanggung jawab yang tumpang tindih, komunikasi yang terlewat, dan jalur eskalasi yang tidak jelas — masalah yang dapat memperburuk kompleksitas teknis yang ada dan mengubah tantangan manajemen sambil meningkatkan risiko pembengkakan waktu dan anggaran.

Contoh matriks RACI berikut adalah titik awal yang dapat membantu Anda memperjelas peran dan tanggung jawab potensial dalam organisasi Anda.

Tugas atau aktivitas	Manajer proyek	Arsitek	Pengembang	Pemangku kepentingan
Identifikasi hasil dan tantangan bisnis	A/R	R	C	–
Tentukan ruang lingkup dan identifikasi persyaratan	A	R	C	C/I
Identifikasi metrik keberhasilan proyek	A	R	C	I
Buat dan jalankan rencana komunikasi	A/R	C	C	I
Tentukan arsitektur target	I	A/R	C	–
Kontrol akses basis data	I	A/R	R	–
Membuat dan melaksana kan rencana kelangsungan bisnis	A/R	C	I	–
Analisis kohesi dan kopling	I	A/R	R	I

Pindahkan logika bisnis (seperti prosedur tersimpan) dari database ke lapisan aplikasi	I	A	R	–
Memisahkan hubungan tabel, yang dikenal sebagai bergabung	I	A	R	–

Mendefinisikan ruang lingkup dan persyaratan untuk dekomposisi database

Ketika Anda menentukan ruang lingkup dan mengidentifikasi persyaratan untuk proyek dekomposisi database Anda, Anda harus bekerja mundur dari kebutuhan organisasi Anda. Ini membutuhkan pendekatan sistematis yang menyeimbangkan kelayakan teknis dengan nilai bisnis. Langkah awal ini menetapkan dasar untuk seluruh proses dan membantu Anda memastikan bahwa tujuan proyek selaras dengan tujuan dan kemampuan organisasi.

Bagian ini berisi topik berikut:

Membangun kerangka analisis inti

Definisi lingkup dimulai dengan alur kerja sistematis yang memandu analisis melalui empat fase yang saling berhubungan. Pendekatan komprehensif ini memastikan bahwa upaya dekomposisi basis data didasarkan pada pemahaman menyeluruh tentang sistem yang ada dan persyaratan operasional. Berikut ini adalah tahapan dalam kerangka analisis inti:

1. Analisis aktor — Mengidentifikasi secara menyeluruh semua sistem dan aplikasi yang berinteraksi dengan database. Ini melibatkan pemetaan kedua produsen yang melakukan operasi tulis dan konsumen yang menangani operasi baca, sambil mendokumentasikan pola akses, frekuensi, dan waktu penggunaan puncak mereka. Tampilan yang berpusat pada pelanggan ini membantu Anda memahami dampak dari setiap perubahan dan mengidentifikasi jalur kritis yang memerlukan perhatian khusus selama dekomposisi.
2. Analisis aktivitas — Selami jauh ke dalam operasi spesifik yang dilakukan setiap aktor. Anda membuat matriks membuat, membaca, memperbarui, dan menghapus (CRUD) terperinci untuk

setiap sistem dan mengidentifikasi tabel mana yang mereka akses dan caranya. Analisis ini membantu Anda menemukan batas-batas alami untuk dekomposisi dan menyoroti area di mana Anda dapat menyederhanakan arsitektur saat ini.

3. Pemetaan ketergantungan - Mendokumentasikan dependensi langsung dan tidak langsung antar sistem, menciptakan visualisasi yang jelas dari aliran data dan hubungan. Ini membantu mengidentifikasi potensi titik putus dan area di mana perencanaan yang matang diperlukan untuk mendapatkan kepercayaan. Analisis mempertimbangkan dependensi teknis, seperti tabel bersama dan kunci asing, dan dependensi proses bisnis, seperti urutan alur kerja dan persyaratan pelaporan.
4. Persyaratan konsistensi - Periksa kebutuhan konsistensi setiap operasi dengan standar tinggi. Tentukan operasi mana yang membutuhkan konsistensi langsung, seperti transaksi keuangan. Operasi lain dapat beroperasi dengan konsistensi akhirnya, seperti pembaruan analitik. Analisis ini secara langsung mempengaruhi pilihan pola dekomposisi dan keputusan arsitektur di seluruh proyek.

Mendefinisikan batas-batas sistem untuk dekomposisi database

Batas sistem adalah perimeter logis yang menentukan di mana satu sistem berakhir dan yang lain dimulai, mencakup kepemilikan data, pola akses, dan titik integrasi. Saat mendefinisikan batas-batas sistem, buatlah pilihan yang bijaksana tetapi menentukan yang menyeimbangkan perencanaan komprehensif dengan kebutuhan implementasi praktis. Pertimbangkan database sebagai unit logis yang mungkin menjangkau beberapa database fisik atau skema. Definisi batas ini mencapai tujuan kritis berikut:

- Mengidentifikasi semua aktor eksternal dan pola interaksinya
- Memetakan dependensi masuk dan keluar secara komprehensif
- Dokumen kendala teknis dan operasional
- Jelas menggambarkan ruang lingkup upaya dekomposisi

Mempertimbangkan siklus rilis

Memahami siklus rilis sangat penting untuk perencanaan dekomposisi basis data. Tinjau waktu pembaruan untuk sistem target dan sistem dependen apa pun. Identifikasi peluang untuk perubahan terkoordinasi. Pertimbangkan penonaktifan sistem terhubung yang direncanakan karena ini dapat memengaruhi strategi dekomposisi Anda. Faktor jendela perubahan yang ada dan kendala

penerapan untuk meminimalkan gangguan bisnis. Pastikan rencana implementasi Anda selaras dengan jadwal rilis di semua sistem yang terhubung.

Mengevaluasi kendala teknis untuk dekomposisi basis data

Sebelum melanjutkan dengan dekomposisi database, menilai keterbatasan teknis utama yang akan membentuk pendekatan modernisasi Anda. Periksa kemampuan tumpukan teknologi Anda saat ini, termasuk versi database, kerangka kerja, persyaratan kinerja, dan perjanjian tingkat layanan. Pertimbangkan mandat keamanan dan kepatuhan, terutama untuk industri yang diatur. Tinjau volume data saat ini, proyeksi pertumbuhan, dan alat migrasi yang tersedia untuk menginformasikan keputusan penskalaan Anda. Terakhir, konfirmasikan hak akses Anda ke kode sumber dan modifikasi sistem karena ini akan menentukan strategi dekomposisi yang layak.

Memahami konteks organisasi

Dekomposisi basis data yang berhasil mengharuskan Anda memahami lanskap organisasi yang lebih luas di mana sistem beroperasi. Memetakan dependensi lintas departemen, dan membangun saluran komunikasi yang jelas antar tim. Nilai kemampuan teknis tim Anda, dan identifikasi kebutuhan pelatihan atau kesenjangan keterampilan yang perlu Anda atasi. Pertimbangkan implikasi manajemen perubahan, termasuk bagaimana mengelola transisi dan menjaga kelangsungan bisnis. Mengevaluasi sumber daya yang tersedia dan kendala apa pun, seperti keterbatasan anggaran atau kepegawaian. Terakhir, selaraskan strategi dekomposisi Anda dengan harapan dan prioritas pemangku kepentingan untuk mempromosikan dukungan berkelanjutan di seluruh proyek.

Menilai risiko dekomposisi basis data

Penilaian risiko yang komprehensif sangat penting untuk keberhasilan dekomposisi basis data. Evaluasi risiko dengan cermat, seperti integritas data selama migrasi, potensi penurunan kinerja sistem, kemungkinan kegagalan integrasi, dan kerentanan keamanan. Tantangan teknis ini harus diimbangi dengan risiko bisnis, termasuk potensi gangguan operasional, keterbatasan sumber daya, penundaan waktu, dan kendala anggaran. Untuk setiap risiko yang teridentifikasi, kembangkan strategi mitigasi spesifik dan rencana darurat untuk mempertahankan momentum proyek sekaligus melindungi operasi bisnis.

Buat matriks risiko yang mengevaluasi dampak dan probabilitas masalah potensial. Bekerja dengan tim teknis dan pemangku kepentingan bisnis untuk mengidentifikasi risiko, menetapkan ambang batas yang jelas untuk intervensi, dan mengembangkan strategi mitigasi khusus. Misalnya, menilai

risiko kehilangan data sebagai dampak tinggi dan probabilitas rendah, dan itu membutuhkan strategi cadangan yang kuat. Degradasi kinerja kecil mungkin dampak sedang dan probabilitas tinggi, dan membutuhkan pemantauan proaktif.

Tetapkan siklus tinjauan risiko reguler untuk menilai kembali prioritas dan menyesuaikan rencana mitigasi saat proyek berkembang. Pendekatan sistematis ini memastikan bahwa sumber daya difokuskan pada risiko paling kritis sambil mempertahankan jalur eskalasi yang jelas untuk masalah yang muncul.

Mendefinisikan kriteria keberhasilan untuk dekomposisi basis data

Kriteria keberhasilan untuk dekomposisi basis data harus didefinisikan dengan jelas dan dapat diukur di berbagai dimensi. Dari perspektif bisnis, tetapkan target spesifik untuk pengurangan biaya, peningkatan time-to-market, ketersediaan sistem, dan kepuasan pelanggan. Keberhasilan teknis harus diukur melalui peningkatan yang dapat diukur dalam kinerja sistem, efisiensi penyebaran, konsistensi data, dan keandalan keseluruhan. Untuk proses migrasi, tentukan persyaratan ketat untuk nol kehilangan data, batas gangguan bisnis yang dapat diterima, kepatuhan anggaran, dan kepatuhan garis waktu.

Dokumentasikan kriteria ini secara menyeluruh dengan mempertahankan metrik dasar dan target, metodologi pengukuran yang jelas, dan jadwal tinjauan reguler. Tetapkan pemilik yang jelas untuk setiap metrik keberhasilan, dan petakan dependensi di antara metrik yang berbeda. Pendekatan komprehensif untuk mengukur keberhasilan ini menyelaraskan pencapaian teknis dengan hasil bisnis, sambil mempertahankan akuntabilitas selama perjalanan dekomposisi.

Mengontrol akses database selama dekomposisi

Banyak organisasi menghadapi skenario umum: database pusat yang telah tumbuh secara organik selama bertahun-tahun dan diakses langsung oleh berbagai layanan dan tim. Ini menciptakan beberapa masalah kritis:

- Pertumbuhan yang tidak terkendali — Ketika tim terus menambahkan fitur baru dan memodifikasi skema, basis data menjadi semakin kompleks dan sulit dikelola.
- Masalah kinerja — Bahkan dengan peningkatan perangkat keras, beban yang meningkat pada akhirnya mengancam untuk melampaui kemampuan database. Ketidakmungkinan untuk menyetel kueri karena kompleksitas skema atau kurangnya keterampilan. Tidak dapat memprediksi atau menjelaskan kinerja sistem.
- Kelumpuhan dekomposisi — Menjadi hampir tidak mungkin untuk membagi atau memfaktorkan ulang database saat sedang dimodifikasi secara aktif oleh beberapa tim.

> **Note**
>
> Sistem database monolitik sering menggunakan kembali kredensi yang sama untuk aplikasi atau layanan atau untuk administrasi. Hal ini menyebabkan ketertelusuran basis data yang buruk. Menetapkan peran khusus dan mengadopsi prinsip hak istimewa terkecil dapat membantu Anda meningkatkan keamanan dan ketersediaan.

Ketika berhadapan dengan database monolitik yang telah menjadi berat, salah satu pola yang paling efektif untuk mengontrol akses disebut layanan pembungkus database. Ini memberikan langkah pertama yang strategis dalam mengelola sistem database yang kompleks. Ini menetapkan akses database terkontrol dan memungkinkan modernisasi bertahap, sekaligus mengurangi risiko. Pendekatan ini menciptakan dasar untuk peningkatan inkremental dengan memberikan visibilitas yang jelas ke dalam pola penggunaan data dan dependensi. Ini adalah arsitektur transisi yang berfungsi sebagai langkah menuju dekomposisi database penuh. Layanan pembungkus memberikan stabilitas dan kontrol yang diperlukan untuk membuat perjalanan itu berhasil.

Bagian ini berisi topik berikut:

- Mengontrol akses dengan pola layanan pembungkus database
- Mengontrol akses dengan pola CQRS

Mengontrol akses dengan pola layanan pembungkus database

Layanan pembungkus adalah lapisan layanan yang bertindak sebagai fasad untuk database. Pendekatan ini sangat berharga ketika Anda perlu mempertahankan fungsionalitas yang ada sambil mempersiapkan dekomposisi masa depan. Pola ini mengikuti prinsip sederhana—ketika sesuatu terlalu berantakan, mulailah dengan menahan kekacauan. Layanan pembungkus menjadi satu-satunya cara resmi untuk mengakses database, menyediakan antarmuka yang terkontrol sambil menyembunyikan kompleksitas yang mendasarinya.

Gunakan pola ini ketika dekomposisi database langsung tidak layak karena skema yang kompleks atau ketika beberapa layanan memerlukan akses data berkelanjutan. Ini sangat berharga selama periode transisi karena menyediakan waktu untuk refactoring hati-hati sambil menjaga stabilitas sistem. Pola ini berfungsi dengan baik saat mengkonsolidasikan kepemilikan data ke tim tertentu atau saat aplikasi baru membutuhkan tampilan agregat di beberapa tabel.

Misalnya, terapkan pola ini ketika:

- Kompleksitas skema mencegah pemisahan segera
- Beberapa tim membutuhkan akses data yang berkelanjutan
- Modernisasi bertahap lebih disukai
- Restrukturisasi tim membutuhkan kepemilikan data yang jelas
- Aplikasi baru membutuhkan tampilan data terkonsolidasi

Manfaat dan keterbatasan pola layanan pembungkus database

Berikut ini adalah manfaat dari pola pembungkus database:

- Pertumbuhan terkontrol - Layanan pembungkus mencegah penambahan lebih lanjut yang tidak terkontrol ke skema database.
- Batas yang jelas — Proses implementasi membantu Anda menetapkan batasan kepemilikan dan tanggung jawab yang jelas.
- Kebebasan refactoring — Layanan pembungkus memungkinkan Anda membuat perubahan internal tanpa memengaruhi konsumen.
- Peningkatan observabilitas - Layanan pembungkus adalah satu titik untuk pemantauan dan pencatatan.

- Pengujian yang disederhanakan - Layanan pembungkus memudahkan penggunaan layanan untuk membuat versi tiruan yang disederhanakan untuk pengujian.

Berikut ini adalah keterbatasan pola pembungkus database.

- Kopling teknologi — Layanan pembungkus bekerja paling baik ketika menggunakan tumpukan teknologi yang sama dengan layanan konsumsi.
- Overhead awal — Layanan pembungkus memerlukan infrastruktur tambahan yang dapat memengaruhi kinerja.
- Upaya migrasi — Untuk mengimplementasikan layanan pembungkus, Anda harus berkoordinasi di seluruh tim untuk beralih dari akses langsung.
- Kinerja - Jika layanan pembungkus mengalami lalu lintas tinggi, penggunaan berat, atau akses yang sering, layanan yang dikonsumsi mungkin mengalami kinerja yang buruk. Di atas database, layanan wrapper harus menangani pagination, kursor, dan koneksi database. Bergantung pada kasus penggunaan Anda, skalanya mungkin tidak baik, dan mungkin kurang cocok untuk beban kerja ekstrak, transformasi, dan beban (ETL).

Menerapkan pola layanan pembungkus database

Ada dua fase untuk mengimplementasikan pola layanan pembungkus database. Pertama, Anda membuat layanan pembungkus database. Kemudian, Anda mengarahkan semua akses melaluinya dan mendokumentasikan pola akses.

Fase 1: Membuat layanan pembungkus database

Buat layer layanan ringan yang bertindak sebagai penjaga gerbang ke database Anda. Awalnya, itu harus mencerminkan semua fungsi yang ada. Layanan pembungkus ini menjadi titik akses wajib untuk semua operasi database, yang mengubah dependensi database langsung menjadi dependensi tingkat layanan. Terapkan pencatatan dan pemantauan terperinci pada lapisan ini untuk melacak pola penggunaan, metrik kinerja, dan frekuensi akses. Pertahankan prosedur tersimpan yang ada, tetapi pastikan bahwa mereka hanya diakses melalui antarmuka layanan baru ini.

Tahap 2: Menerapkan kontrol akses

Secara sistematis mengalihkan semua akses database melalui layanan pembungkus, dan kemudian mencabut izin database langsung dari sistem eksternal yang mengakses database secara langsung. Dokumentasikan setiap pola akses dan ketergantungan saat layanan dimigrasikan. Akses terkontrol

ini memungkinkan refactoring internal komponen database tanpa mengganggu konsumen eksternal. Misalnya, mulailah dengan risiko rendah, operasi hanya-baca alih-alih alur kerja transaksional yang kompleks.

Fase 3: Memantau kinerja database

Gunakan layanan pembungkus sebagai titik pemantauan terpusat untuk kinerja database. Lacak metrik kunci, termasuk waktu respons kueri, pola penggunaan, tingkat kesalahan, dan pemanfaatan sumber daya. Siapkan peringatan untuk ambang kinerja dan pola yang tidak biasa. Misalnya, pantau kueri yang berjalan lambat, pemanfaatan kumpulan koneksi, dan throughput transaksi untuk secara proaktif mengidentifikasi potensi masalah.

Gunakan tampilan konsolidasi ini untuk mengoptimalkan kinerja database melalui penyetelan kueri, penyesuaian alokasi sumber daya, dan analisis pola penggunaan. Sifat terpusat dari layanan pembungkus membuatnya lebih mudah untuk menerapkan perbaikan dan memvalidasi dampaknya di semua konsumen, sambil mempertahankan standar kinerja yang konsisten.

Praktik terbaik untuk mengimplementasikan layanan pembungkus database

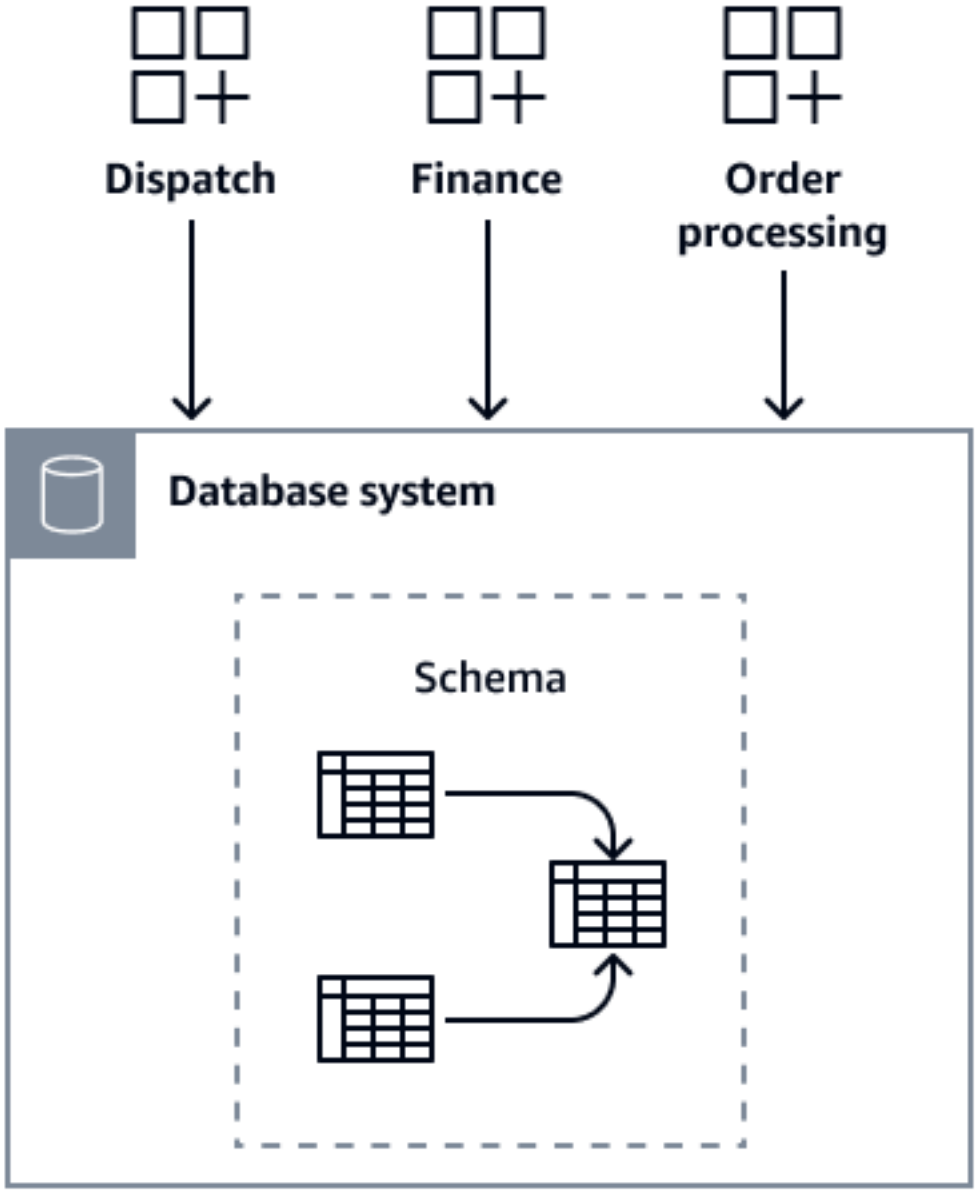
Praktik terbaik berikut dapat membantu Anda mengimplementasikan layanan pembungkus database:

- Mulai dari yang kecil - Mulailah dengan pembungkus minimal yang hanya memproksi fungsionalitas yang ada
- Menjaga stabilitas - Menjaga antarmuka layanan tetap stabil saat melakukan perbaikan internal
- Memantau penggunaan - Menerapkan pemantauan komprehensif untuk memahami pola akses
- Kepemilikan yang jelas - Tetapkan tim khusus untuk mempertahankan pembungkus dan skema yang mendasarinya
- Mendorong penyimpanan lokal — Memotivasi tim untuk menyimpan data mereka di database mereka sendiri

Contoh berbasis skenario

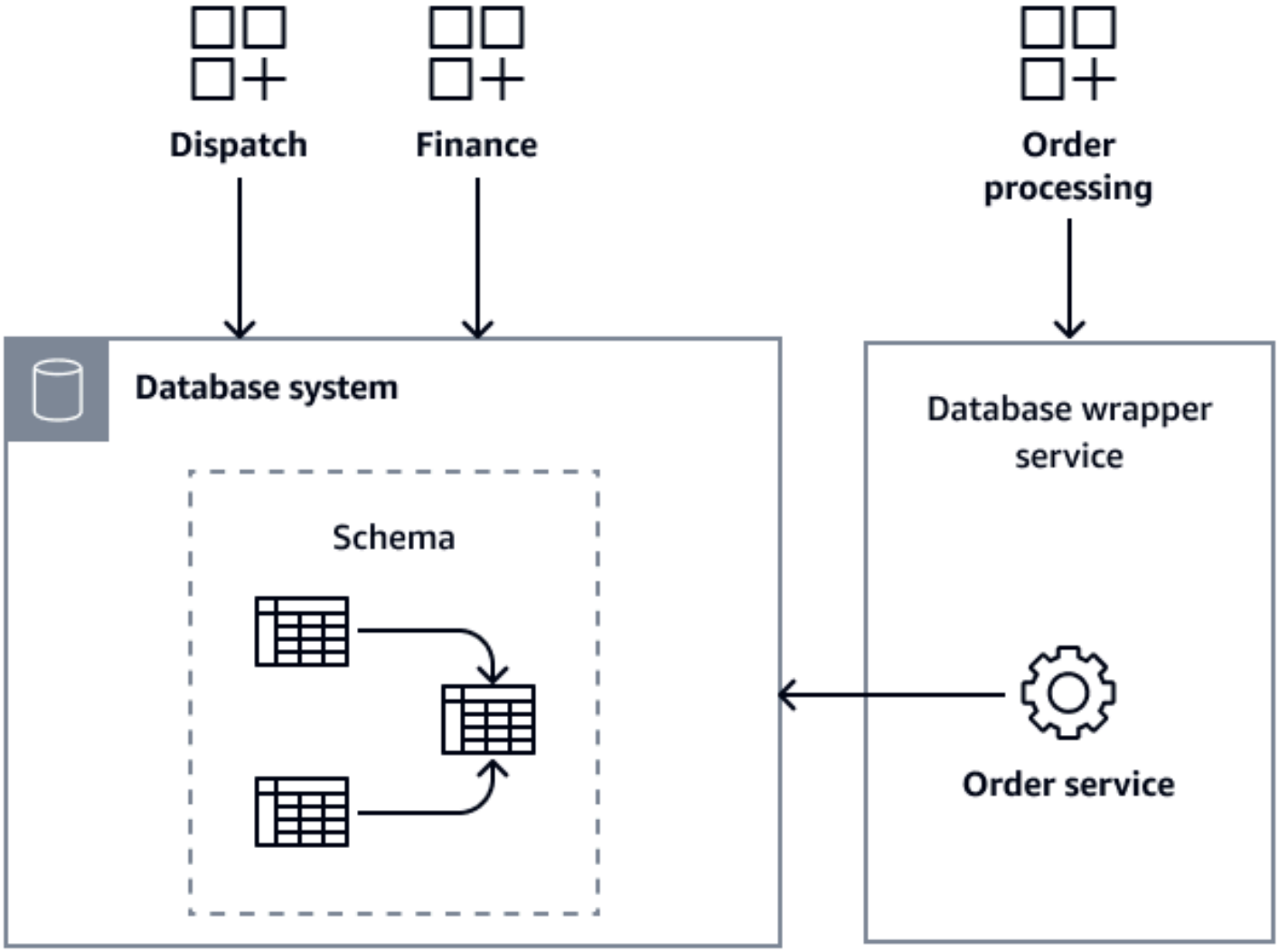
Bagian ini menjelaskan contoh bagaimana perusahaan fiktif, bernama AnyCompany Books, dapat menggunakan pola pembungkus database untuk mengontrol akses ke sistem database monolitik mereka. Di AnyCompany Books, ada tiga layanan penting: Pengiriman, Keuangan, dan Pemrosesan Pesanan. Layanan ini berbagi akses ke database pusat. Setiap layanan dikelola oleh tim yang berbeda. Seiring waktu, mereka secara independen memodifikasi skema database untuk memenuhi

kebutuhan spesifik mereka. Hal ini menyebabkan jaringan dependensi yang kusut dan struktur basis data yang semakin kompleks.



Aplikasi perusahaan atau arsitek perusahaan mengakui perlunya menguraikan basis data monolitik ini. Tujuan mereka adalah untuk memberikan setiap layanan database khusus untuk meningkatkan pemeliharaan dan mengurangi ketergantungan lintas tim. Namun, mereka menghadapi tantangan yang signifikan—hampir tidak mungkin untuk menguraikan database sementara ketiga tim terus memodifikasinya secara aktif untuk proyek mereka yang sedang berlangsung. Perubahan skema yang konstan dan kurangnya koordinasi antar tim membuatnya sangat berisiko untuk mencoba restrukturisasi yang signifikan.

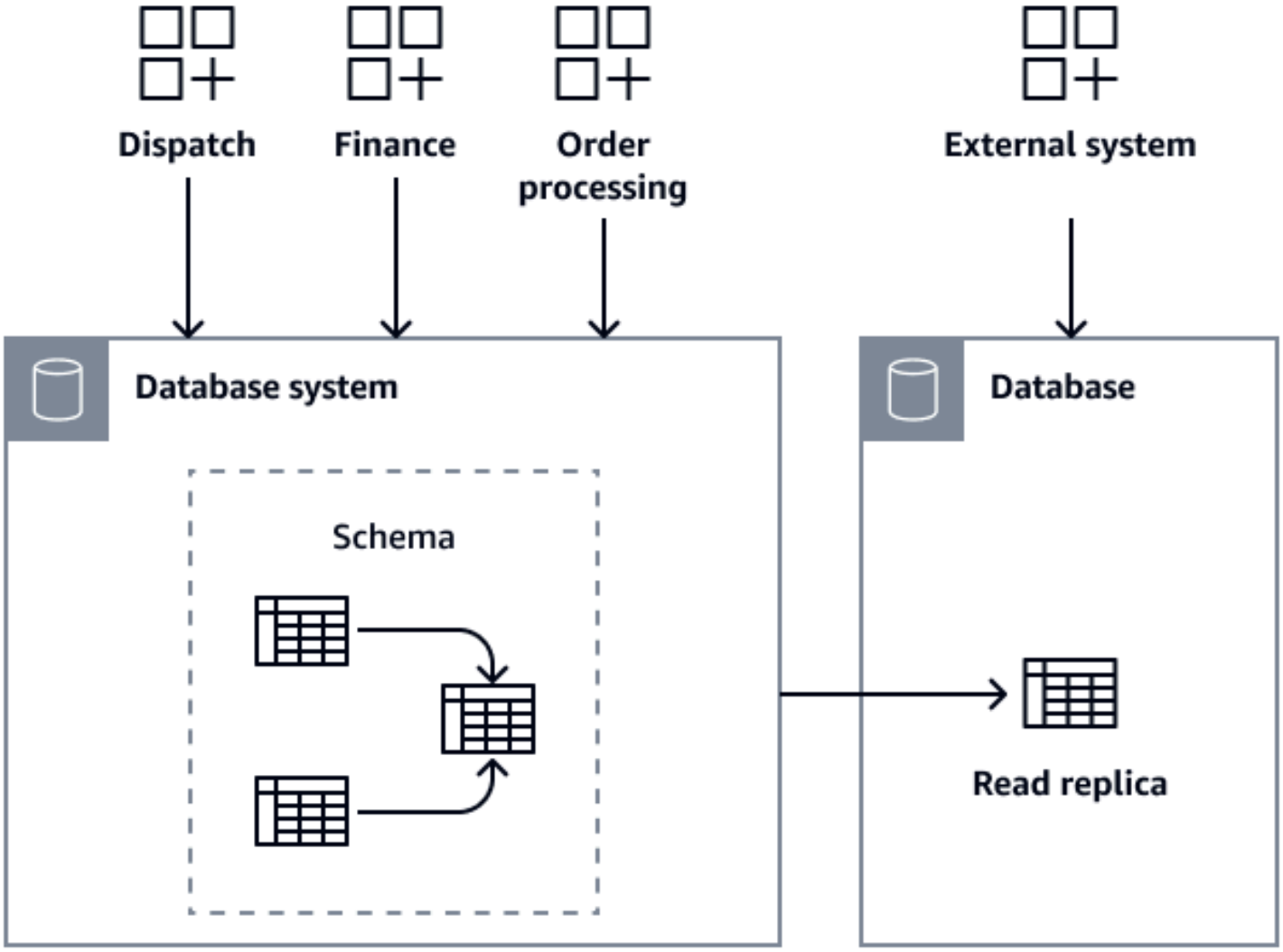
Arsitek menggunakan pola layanan pembungkus database untuk mulai mengontrol akses ke database monolitik. Pertama, mereka menyiapkan layanan pembungkus database untuk modul tertentu, yang disebut layanan Order. Kemudian, mereka mengarahkan layanan Pemrosesan Pesanan untuk mengakses layanan pembungkus alih-alih mengakses database secara langsung. Gambar berikut menunjukkan infrastruktur yang dimodifikasi.



Mengontrol akses dengan pola CQRS

Pola lain yang dapat Anda gunakan untuk mengisolasi sistem eksternal yang terhubung ke database pusat ini adalah pemisahan tanggung jawab permintaan perintah (CQRS). Jika beberapa sistem eksternal terhubung ke database pusat Anda terutama untuk dibaca, seperti analitik, pelaporan, atau operasi intensif baca lainnya, Anda dapat membuat penyimpanan data yang dioptimalkan untuk dibaca secara terpisah.

Pola ini secara efektif mengisolasi sistem eksternal ini dari dampak dekomposisi database dan perubahan skema. Dengan mempertahankan replika baca khusus atau penyimpanan data yang dibuat khusus untuk pola kueri tertentu, tim dapat melanjutkan operasinya tanpa terpengaruh oleh perubahan dalam struktur basis data utama. Misalnya, saat Anda menguraikan basis data monolitik Anda, sistem pelaporan dapat terus bekerja dengan tampilan data yang ada, dan beban kerja analitis dapat mempertahankan pola kueri mereka saat ini melalui penyimpanan analitik khusus. Pendekatan ini memberikan isolasi teknis dan memungkinkan otonomi organisasi karena tim yang berbeda dapat mengembangkan sistem mereka secara independen tanpa keterkaitan erat dengan perjalanan transformasi database utama.



Untuk informasi lebih lanjut tentang pola ini dan contoh penggunaannya untuk memisahkan hubungan tabel, lihat Pola CQRS nanti dalam panduan ini.

Menganalisis kohesi dan kopling untuk dekomposisi basis data

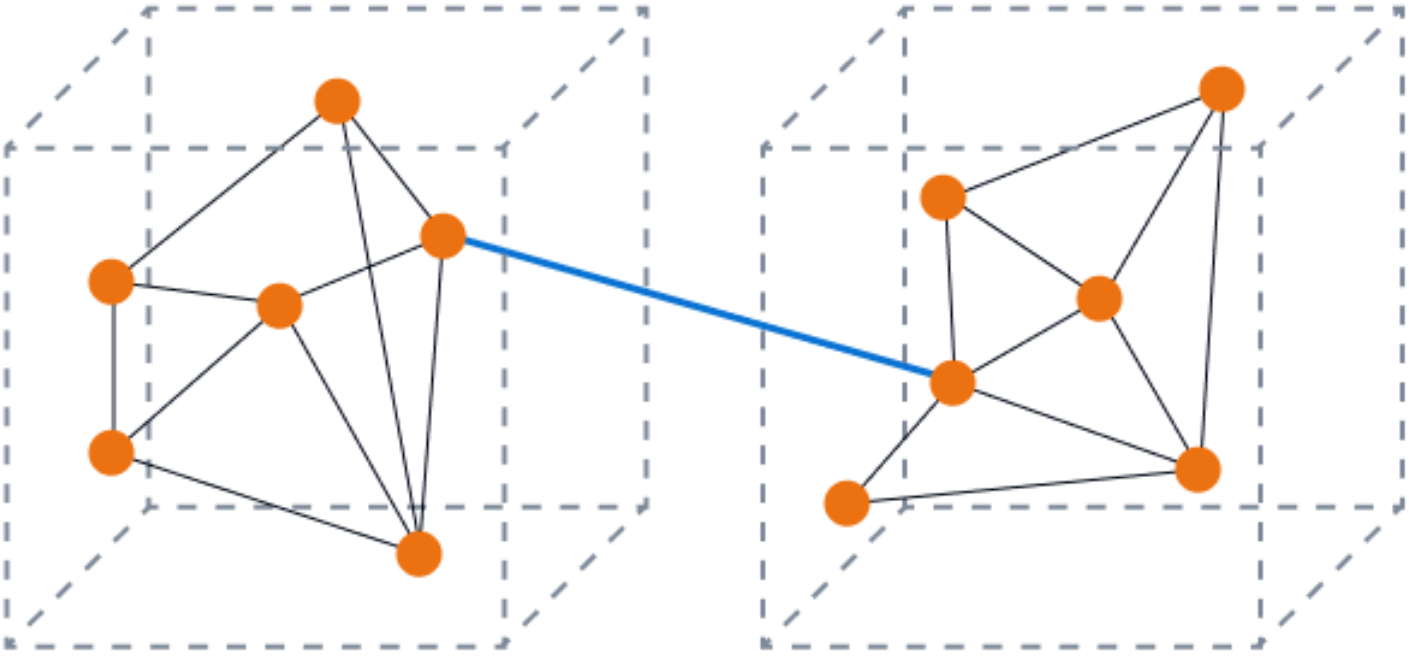
Bagian ini membantu Anda menganalisis pola kopling dan kohesi dalam database monolitik Anda untuk memandu dekomposisinya. Memahami bagaimana komponen database berinteraksi dan bergantung satu sama lain sangat penting untuk mengidentifikasi titik istirahat alami, menilai kompleksitas, dan merencanakan pendekatan migrasi bertahap. Analisis ini mengungkapkan ketergantungan tersembunyi, menyoroti area yang cocok untuk pemisahan segera, dan membantu Anda memprioritaskan upaya dekomposisi sambil meminimalkan risiko transformasi. Dengan memeriksa kopling dan kohesi, Anda dapat membuat keputusan berdasarkan informasi tentang urutan pemisahan komponen untuk menjaga stabilitas sistem selama proses transformasi.

Bagian ini berisi topik berikut:

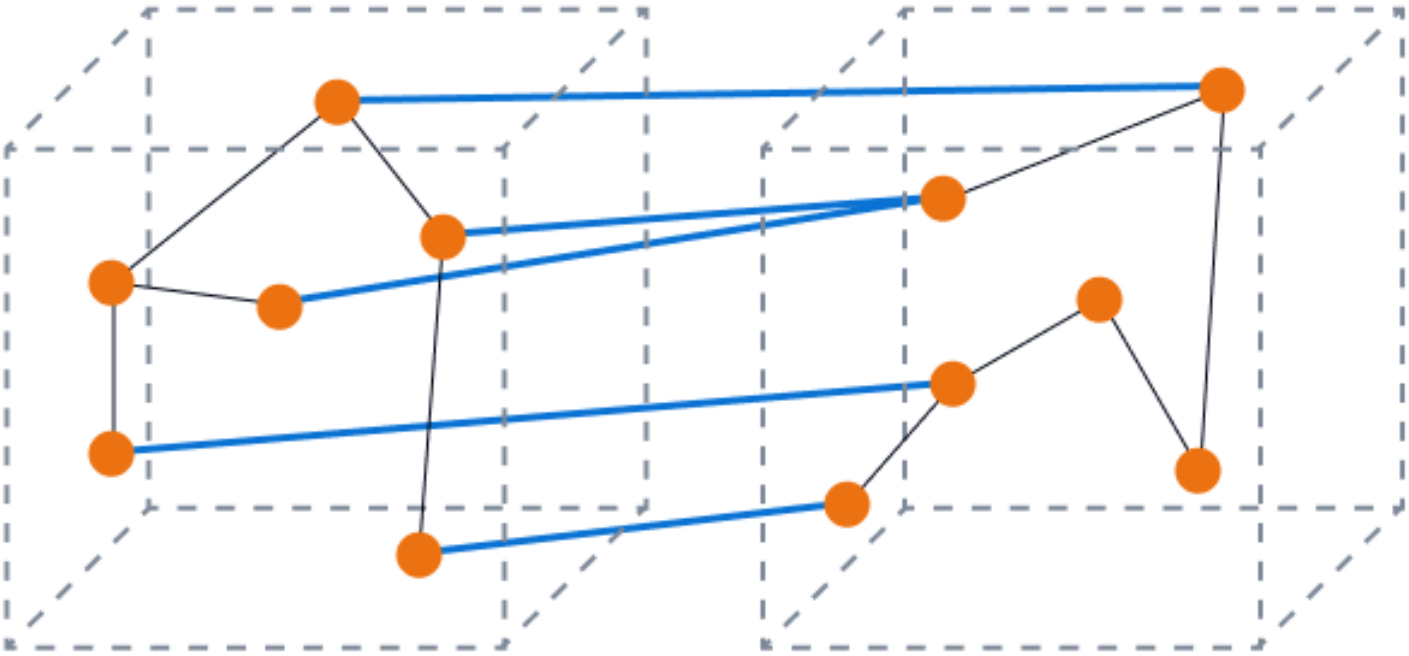
Tentang kohesi dan kopling

Kopling mengukur tingkat saling ketergantungan antara komponen database. Dalam sistem yang dirancang dengan baik, Anda ingin mencapai kopling longgar, di mana perubahan pada satu komponen memiliki dampak minimal pada komponen lain. Kohesi mengukur seberapa baik elemen-elemen dalam komponen database bekerja sama untuk melayani satu tujuan yang terdefinisi dengan baik. Kohesi yang tinggi menunjukkan bahwa elemen komponen sangat terkait dan berfokus pada fungsi tertentu. Saat menguraikan database monolitik, Anda harus menganalisis kohesi dalam masing-masing komponen dan kopling di antara keduanya. Analisis ini membantu Anda membuat keputusan berdasarkan informasi tentang cara memecah database sambil mempertahankan integritas dan kinerja sistem.

Gambar berikut menunjukkan kopling longgar dengan kohesi tinggi. Komponen dalam database bekerja sama untuk melakukan fungsi tertentu, dan Anda meminimalkan dampak perubahan pada satu komponen. Ini adalah keadaan ideal.



Gambar berikut menunjukkan kopling tinggi dengan kohesi rendah. Komponen database terputus, dan perubahan sangat mungkin berdampak pada komponen lain.



Pola kopling umum dalam database monolitik

Ada beberapa pola kopling yang biasa ditemukan saat menguraikan database monolitik menjadi database khusus layanan mikro. Memahami pola-pola ini sangat penting untuk inisiatif modernisasi

database yang sukses. Bagian ini menjelaskan setiap pola, tantangannya, dan praktik terbaik untuk mengurangi kopling.

Pola kopling implementasi

Definisi: Komponen saling berhubungan erat pada tingkat kode dan skema. Misalnya, memodifikasi struktur `customer` tabel berdampak`order`,`inventory`, dan `billing` layanan.

Dampak modernisasi: Setiap layanan mikro memerlukan skema database khusus dan lapisan akses datanya sendiri.

Tantangan:

- Perubahan pada tabel bersama memengaruhi beberapa layanan
- Risiko tinggi efek samping yang tidak diinginkan
- Peningkatan kompleksitas pengujian
- Sulit untuk memodifikasi komponen individual

Praktik terbaik untuk mengurangi kopling:

- Tentukan antarmuka yang jelas antar komponen
- Gunakan lapisan abstraksi untuk menyembunyikan detail implementasi
- Menerapkan skema khusus domain

Pola kopling temporal

Definisi: Operasi harus berjalan dalam urutan tertentu. Misalnya, pemrosesan pesanan tidak dapat dilanjutkan hingga pembaruan inventaris selesai.

Dampak modernisasi: Setiap layanan mikro membutuhkan kontrol data otonom.

Tantangan:

- Memecah dependensi sinkron antar layanan
- Kemacetan kinerja
- Sulit untuk mengoptimalkan

- Pemrosesan paralel terbatas

Praktik terbaik untuk mengurangi kopling:

- Menerapkan pemrosesan asinkron jika memungkinkan
- Gunakan arsitektur berbasis peristiwa
- Desain untuk konsistensi akhirnya bila sesuai

Pola kopling penyebaran

Definisi: Komponen sistem harus digunakan sebagai satu unit. Misalnya, perubahan kecil pada logika pemrosesan pembayaran memerlukan pemindahan seluruh database.

Dampak modernisasi: Penerapan basis data independen per layanan

Tantangan:

- Penerapan berisiko tinggi
- Frekuensi penyebaran terbatas
- Prosedur rollback yang kompleks

Praktik terbaik untuk mengurangi kopling:

- Memecah menjadi komponen yang dapat diterapkan secara independen
- Menerapkan strategi sharding database
- Gunakan pola penyebaran biru-hijau

Pola kopling domain

Definisi: Domain bisnis berbagi struktur dan logika database. Misalnya, `customerorder`, dan `inventory` domain berbagi tabel dan prosedur yang disimpan.

Dampak modernisasi: Isolasi data khusus domain

Tantangan:

- Batas domain yang kompleks

- Sulit untuk menskalakan domain individu
- Aturan bisnis kusut

Praktik terbaik untuk mengurangi kopling:

- Identifikasi batas domain yang jelas
- Pisahkan data berdasarkan konteks domain
- Menerapkan layanan khusus domain

Pola kohesi umum dalam database monolitik

Ada beberapa pola kohesi yang biasa ditemukan ketika mengevaluasi komponen database untuk dekomposisi. Memahami pola-pola ini sangat penting untuk mengidentifikasi komponen database yang terstruktur dengan baik. Bagian ini menjelaskan setiap pola, karakteristiknya, dan praktik terbaik untuk memperkuat kohesi.

Pola kohesi fungsional

Definisi: Semua elemen secara langsung mendukung dan berkontribusi untuk melakukan fungsi tunggal yang terdefinisi dengan baik. Misalnya, semua prosedur dan tabel yang disimpan dalam modul pemrosesan pembayaran hanya menangani operasi terkait pembayaran.

Dampak modernisasi: Pola ideal untuk desain database microservice

Tantangan:

- Mengidentifikasi batas-batas fungsional yang jelas
- Memisahkan komponen tujuan campuran
- Mempertahankan tanggung jawab tunggal

Praktik terbaik untuk memperkuat kohesi:

- Kelompokkan fungsi terkait bersama
- Hapus fungsionalitas yang tidak terkait
- Tentukan batas komponen yang jelas

Pola kohesi berurutan

Definisi: Output dari satu elemen menjadi masukan untuk elemen lainnya. Misalnya, hasil validasi untuk umpan pesanan ke pemrosesan pesanan.

Dampak modernisasi: Membutuhkan analisis alur kerja yang cermat dan pemetaan aliran data

Tantangan:

- Mengelola dependensi antar langkah
- Menangani skenario kegagalan
- Mempertahankan ketertiban proses

Praktik terbaik untuk memperkuat kohesi:

- Dokumentasikan aliran data yang jelas
- Menerapkan penanganan kesalahan yang tepat
- Desain antarmuka yang jelas di antara langkah-langkah

Pola kohesi komunikasi

Definisi: Elemen beroperasi pada data yang sama. Misalnya, fungsi manajemen profil pelanggan semuanya bekerja dengan data pelanggan.

Dampak modernisasi: Membantu mengidentifikasi batas data untuk pemisahan layanan guna mengurangi kopling antar modul

Tantangan:

- Menentukan kepemilikan data
- Mengelola akses data bersama
- Menjaga konsistensi data

Praktik terbaik untuk memperkuat kohesi:

- Tentukan kepemilikan data yang jelas
- Menerapkan pola akses data yang tepat

- Desain partisi data yang efektif

Pola kohesi prosedural

Definisi: Elemen dikelompokkan bersama karena mereka harus dieksekusi dalam urutan tertentu, tetapi mereka mungkin tidak terkait secara fungsional. Misalnya, dalam pemrosesan pesanan, prosedur tersimpan yang menangani validasi pesanan dan pemberitahuan pengguna dikelompokkan bersama hanya karena terjadi secara berurutan, meskipun mereka melayani tujuan yang berbeda dan dapat ditangani oleh layanan terpisah.

Dampak modernisasi: Membutuhkan pemisahan prosedur yang cermat sambil mempertahankan aliran proses

Tantangan:

- Mempertahankan aliran proses yang benar setelah dekomposisi
- Mengidentifikasi batas-batas fungsional yang sebenarnya dibandingkan dengan dependensi prosedural

Praktik terbaik untuk memperkuat kohesi:

- Prosedur terpisah berdasarkan tujuan fungsionalnya daripada perintah eksekusi
- Gunakan pola orkestrasi untuk mengelola alur proses
- Menerapkan sistem manajemen alur kerja untuk urutan yang kompleks
- Rancang arsitektur berbasis peristiwa untuk menangani langkah-langkah proses secara mandiri

Pola kohesi temporal

Definisi: Elemen terkait dengan persyaratan waktu. Misalnya, ketika pesanan ditempatkan, beberapa operasi harus dijalankan bersama: pemeriksaan inventaris, pemrosesan pembayaran, konfirmasi pesanan, dan pemberitahuan pengiriman semuanya harus dilakukan dalam jangka waktu tertentu untuk mempertahankan status pesanan yang konsisten.

Dampak modernisasi: Mungkin memerlukan penanganan khusus dalam sistem terdistribusi

Tantangan:

- Mengkoordinasikan dependensi waktu di seluruh layanan terdistribusi
- Mengelola transaksi terdistribusi
- Mengonfirmasi penyelesaian proses di beberapa komponen

Praktik terbaik untuk memperkuat kohesi:

- Menerapkan mekanisme penjadwalan dan batas waktu yang tepat
- Gunakan arsitektur berbasis peristiwa dengan penanganan urutan yang jelas
- Desain untuk konsistensi akhirnya dengan pola kompensasi
- Menerapkan pola saga untuk transaksi terdistribusi

Pola kohesi logis atau kebetulan

Definisi: Elemen secara logis dikategorikan untuk melakukan hal yang sama, meskipun mereka memiliki hubungan yang lemah atau tidak bermakna. Contohnya adalah menyimpan data pesanan pelanggan, jumlah inventaris gudang, dan template email pemasaran dalam skema database yang sama karena semuanya berhubungan dengan operasi penjualan, meskipun memiliki pola akses yang berbeda, manajemen siklus hidup, dan persyaratan penskalaan. Contoh lain adalah menggabungkan pemrosesan pembayaran pesanan dan manajemen katalog produk dalam komponen database yang sama karena keduanya merupakan bagian dari sistem e-commerce, meskipun mereka melayani fungsi bisnis yang berbeda dengan kebutuhan operasional yang berbeda.

Dampak modernisasi: Harus difaktorkan ulang atau direorganisasi

Tantangan:

- Mengidentifikasi pola organisasi yang lebih baik
- Memecah dependensi yang tidak perlu
- Restrukturisasi komponen yang dikelompokkan secara sewenang-wenang

Praktik terbaik untuk memperkuat kohesi:

- Mengatur ulang berdasarkan batas-batas fungsional yang sebenarnya dan domain bisnis
- Hapus pengelompokan sewenang-wenang berdasarkan hubungan dangkal
- Menerapkan pemisahan elemen yang tepat berdasarkan kemampuan bisnis

- Sejajarkan komponen database dengan persyaratan operasional spesifiknya

Menerapkan kopling rendah dan kohesi tinggi

Praktik terbaik

Praktik terbaik berikut dapat membantu Anda mencapai kopling rendah:

- Minimalkan dependensi antar komponen database
- Gunakan antarmuka yang terdefinisi dengan baik untuk interaksi komponen
- Hindari status bersama dan struktur data global

Praktik terbaik berikut dapat membantu Anda mencapai kohesi tinggi:

- Kelompokkan data dan operasi terkait bersama
- Pastikan setiap komponen memiliki tanggung jawab tunggal yang jelas
- Pertahankan batasan yang jelas antara domain bisnis yang berbeda

Fase 1: Memetakan dependensi data

Memetakan hubungan data dan mengidentifikasi batas-batas alam. Anda dapat menggunakan alat, seperti SchemaSpy, untuk memvisualisasikan database dengan menampilkan tabel dalam diagram entity-relationship (ER). Ini memberikan analisis statis database dan menunjukkan beberapa batas dan dependensi yang jelas dalam database.

Anda juga dapat mengekspor skema database Anda dalam database grafik atau di Jupiter buku catatan. Kemudian, Anda dapat menerapkan algoritma pengelompokan atau komponen yang saling berhubungan untuk mengidentifikasi batas dan dependensi alami. AWS Partner Alat lain, seperti CAST Imaging, dapat membantu memahami dependensi database Anda.

Tahap 2: Menganalisis batas-batas transaksi dan pola akses

Menganalisis pola transaksi untuk menjaga sifat atomisitas, konsistensi, isolasi, daya tahan (ACID) dan memahami bagaimana data diakses dan dimodifikasi. Anda dapat menggunakan analisis database dan alat diagnosis, seperti Oracle Automatic Workload Repository (AWR)atau PostgreSQL pg_stat_statements. Analisis ini membantu Anda memahami siapa yang mengakses database

dan apa batas transaksi. Ini juga dapat membantu Anda memahami kohesi dan kopling antar tabel saat runtime. Anda juga dapat menggunakan alat pemantauan dan pembuatan profil yang dapat menautkan kode dan profil eksekusi database, seperti Dynatrace AppEngine.

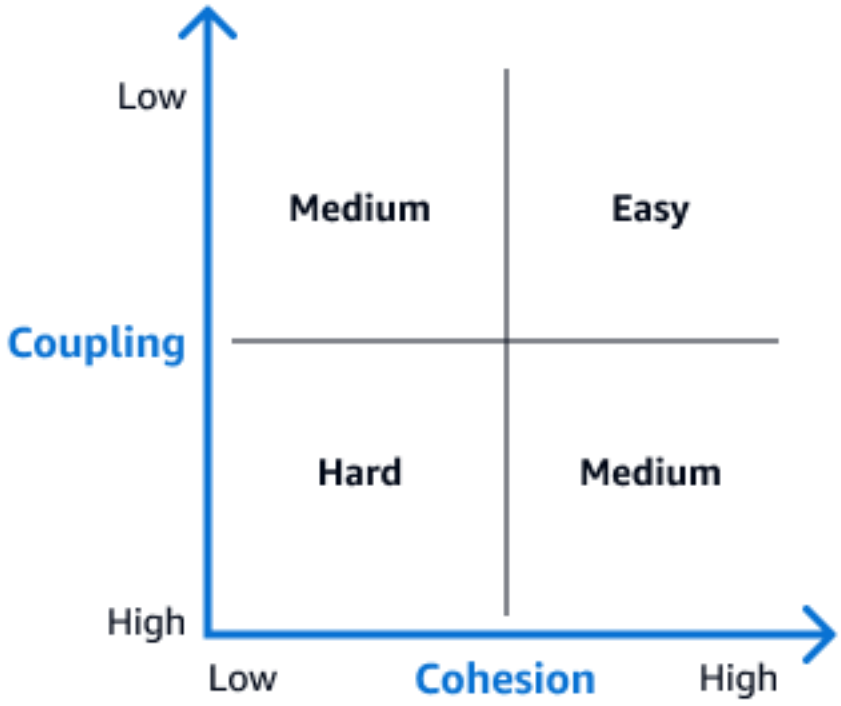
Alat AI, seperti vFunction, dapat membantu Anda mengidentifikasi batas domain dengan menganalisis batas fungsional dan domain aplikasi. Meskipun vFunction terutama menganalisis lapisan aplikasi, wawasannya dapat memandu dekomposisi aplikasi dan database, mendukung penyelarasan dengan domain bisnis.

Fase 3: Identifikasi tabel mandiri

Cari tabel yang menunjukkan dua karakteristik utama:

- Kohesi tinggi — Isi tabel sangat terkait satu sama lain
- Kopling rendah — Mereka memiliki ketergantungan minimal pada tabel lain.

Matriks kohesi kopling berikut dapat membantu Anda mengidentifikasi kesulitan memisahkan setiap tabel. Tabel yang muncul di kuadran kanan atas matriks ini adalah kandidat ideal untuk upaya decoupling awal karena paling mudah dipisahkan. Dalam diagram ER, tabel ini memiliki beberapa hubungan kunci asing atau dependensi lainnya. Setelah Anda memisahkan tabel ini, maju ke tabel dengan hubungan yang lebih kompleks.



> **Note**
>
> Struktur database sering mencerminkan arsitektur aplikasi. Tabel yang lebih mudah dipisahkan pada tingkat database biasanya sesuai dengan komponen yang lebih mudah dikonversi menjadi layanan mikro di tingkat aplikasi.

Migrasi logika bisnis dari database ke lapisan aplikasi

Migrasi logika bisnis dari prosedur, pemicu, dan fungsi yang disimpan database ke layanan lapisan aplikasi merupakan langkah penting dalam menguraikan basis data monolitik. Transformasi ini meningkatkan otonomi layanan, menyederhanakan pemeliharaan, dan meningkatkan skalabilitas. Bagian ini memberikan panduan untuk menganalisis logika database, merencanakan strategi migrasi, dan kemudian menerapkan transformasi sambil mempertahankan kelangsungan bisnis. Ini juga membahas pembentukan rencana rollback yang efektif.

Bagian ini berisi topik berikut:

- Tahap 1: Menganalisis logika bisnis
- Tahap 2: Mengklasifikasikan logika bisnis
- Tahap 3: Migrasi logika bisnis
- Strategi rollback untuk logika bisnis

Tahap 1: Menganalisis logika bisnis

Saat memodernisasi basis data monolitik, Anda harus terlebih dahulu melakukan analisis komprehensif terhadap logika basis data yang ada. Fase ini berfokus pada tiga kategori utama:

- Prosedur tersimpan sering berisi operasi bisnis penting, termasuk logika manipulasi data, aturan bisnis, pemeriksaan validasi, dan perhitungan. Sebagai komponen inti dari logika bisnis aplikasi, mereka memerlukan dekomposisi yang cermat. Misalnya, prosedur tersimpan organisasi keuangan mungkin menangani perhitungan bunga, rekonsiliasi akun, dan pemeriksaan kepatuhan.
- Pemicu adalah komponen basis data utama yang menangani jejak audit, validasi data, perhitungan, dan konsistensi lintas tabel. Misalnya, organisasi ritel mungkin menggunakan pemicu untuk mengelola pembaruan inventaris di seluruh sistem pemrosesan pesanan mereka, yang menunjukkan kompleksitas operasi basis data otomatis.
- Fungsi dalam database terutama mengelola transformasi data, perhitungan, dan operasi pencarian. Mereka sering tertanam di berbagai prosedur dan aplikasi. Misalnya, organisasi perawatan kesehatan mungkin menggunakan fungsi untuk menormalkan data pasien atau mencari kode medis.

Setiap kategori mewakili aspek yang berbeda dari logika bisnis yang tertanam dalam lapisan database. Anda perlu mengevaluasi dan merencanakan masing-masing dengan hati-hati agar memigrasikannya ke lapisan aplikasi.

Selama fase analisis ini, pelanggan biasanya menghadapi tiga tantangan signifikan. Pertama, dependensi kompleks muncul melalui panggilan prosedur bersarang, referensi lintas skema, dan dependensi data implisit. Kedua, manajemen transaksi menjadi penting, terutama ketika berhadapan dengan transaksi multi-langkah dan menjaga konsistensi data di seluruh sistem terdistribusi. Ketiga, pertimbangan kinerja harus dievaluasi dengan cermat, terutama untuk operasi pemrosesan batch, pembaruan data massal, dan perhitungan waktu nyata yang saat ini mendapat manfaat dari kedekatan dengan data.

Untuk mengatasi tantangan ini secara efektif, Anda dapat menggunakan AWS Schema Conversion Tool (AWS SCT) untuk analisis awal dan kemudian menggunakan alat pemetaan ketergantungan terperinci. Pendekatan ini membantu Anda memahami cakupan penuh logika database Anda dan membuat strategi migrasi komprehensif yang menjaga kelangsungan bisnis selama dekomposisi.

Dengan memahami komponen dan tantangan ini secara menyeluruh, Anda dapat merencanakan perjalanan modernisasi Anda dengan lebih baik dan membuat keputusan berdasarkan informasi tentang elemen mana yang harus diprioritaskan selama migrasi ke arsitektur berbasis layanan mikro.

Saat menganalisis komponen kode database, buat dokumentasi komprehensif untuk setiap prosedur, pemicu, dan fungsi yang disimpan. Mulailah dengan menjelaskan dengan jelas tujuan dan fungsionalitas intinya, termasuk aturan bisnis yang diterapkan. Detail semua parameter input dan output, dan catat tipe data dan rentang yang valid. Memetakan dependensi pada objek database lain, sistem eksternal, dan proses hilir. Mendefinisikan dengan jelas batas-batas transaksi dan persyaratan isolasi untuk menjaga integritas data. Dokumentasikan ekspektasi kinerja apa pun, termasuk persyaratan waktu respons dan pola pemanfaatan sumber daya. Terakhir, analisis pola penggunaan untuk memahami beban puncak, frekuensi eksekusi, dan periode bisnis kritis.

Tahap 2: Mengklasifikasikan logika bisnis

Dekomposisi database yang efektif memerlukan kategorisasi sistematis logika database di seluruh dimensi kunci: kompleksitas, dampak bisnis, dependensi, pola penggunaan, dan kesulitan migrasi. Klasifikasi ini membantu Anda mengidentifikasi komponen berisiko tinggi, menentukan persyaratan pengujian, dan menetapkan prioritas migrasi. Misalnya, prosedur tersimpan yang kompleks dengan dampak bisnis yang tinggi dan penggunaan yang sering memerlukan perencanaan yang cermat dan pengujian ekstensif. Namun, fungsi sederhana yang jarang digunakan dengan dependensi minimal mungkin cocok untuk fase migrasi awal.

Pendekatan terstruktur ini menciptakan peta jalan migrasi seimbang yang meminimalkan gangguan bisnis sambil menjaga stabilitas sistem. Dengan memahami hubungan timbal balik ini, Anda dapat meningkatkan urutan upaya dekomposisi Anda dan mengalokasikan sumber daya dengan tepat.

Tahap 3: Migrasi logika bisnis

Setelah Anda menganalisis dan mengklasifikasikan logika bisnis Anda, sekarang saatnya untuk memigrasikannya. Ada dua pendekatan ketika migrasi logika bisnis dari database monolitik: memindahkan logika database ke lapisan aplikasi, atau memindahkan logika bisnis ke database lain yang merupakan bagian dari microservice.

Jika Anda memigrasikan logika bisnis ke aplikasi, maka tabel database hanya menyimpan data, dan database tidak berisi logika bisnis apa pun. Ini adalah pendekatan yang direkomendasikan. Anda dapat menggunakan Ispirer atau alat AI generatif, seperti Amazon Q Developer atau Kiro, untuk mengonversi logika bisnis basis data untuk lapisan aplikasi, seperti konversi ke Java. Untuk informasi selengkapnya, lihat Memigrasikan logika bisnis dari database ke aplikasi untuk inovasi dan fleksibilitas yang lebih cepat (posting AWS blog).

Jika Anda memigrasikan logika bisnis ke database lain, Anda dapat menggunakan AWS Schema Conversion Tool (AWS SCT) untuk mengonversi skema database dan objek kode yang ada ke database target Anda. Ini mendukung layanan AWS database yang dibuat khusus, seperti Amazon DynamoDB, Amazon Aurora, dan Amazon Redshift. Dengan menyediakan laporan penilaian yang komprehensif dan kemampuan konversi otomatis, AWS SCT membantu merampingkan proses transisi, memungkinkan Anda untuk fokus pada pengoptimalan struktur database baru Anda untuk meningkatkan kinerja dan skalabilitas. Saat Anda maju melalui proyek modernisasi Anda, AWS SCT dapat menangani konversi inkremental untuk mendukung pendekatan bertahap, memungkinkan Anda untuk memvalidasi dan menyempurnakan setiap langkah transformasi database Anda.

Strategi rollback untuk logika bisnis

Dua aspek penting dari setiap strategi dekomposisi adalah menjaga kompatibilitas mundur dan menerapkan prosedur rollback yang komprehensif. Elemen-elemen ini bekerja sama untuk membantu melindungi operasi selama masa transisi. Bagian ini menjelaskan cara mengelola kompatibilitas selama proses dekomposisi dan menetapkan kemampuan rollback darurat yang efektif yang melindungi terhadap potensi masalah.

Pertahankan kompatibilitas ke belakang

Selama dekomposisi basis data, menjaga kompatibilitas mundur sangat penting untuk transisi yang lancar. Pertahankan prosedur database yang ada sementara sementara sementara menerapkan fungsionalitas baru secara bertahap. Gunakan kontrol versi untuk melacak semua perubahan dan mengelola beberapa versi database secara bersamaan. Merencanakan periode koeksistensi yang diperpanjang di mana sumber dan sistem target harus beroperasi dengan andal. Ini memberikan waktu untuk menguji dan memvalidasi sistem baru sebelum menghentikan komponen lama. Pendekatan ini meminimalkan gangguan bisnis dan menyediakan jaring pengaman untuk rollback jika diperlukan.

Rencana rollback darurat

Strategi rollback yang komprehensif sangat penting untuk dekomposisi basis data yang aman. Terapkan flag fitur dalam kode Anda untuk mengontrol versi logika bisnis mana yang aktif. Ini memungkinkan Anda untuk langsung beralih antara implementasi baru dan asli tanpa perubahan penerapan. Pendekatan ini memberikan kontrol halus atas transisi dan membantu Anda memutar kembali dengan cepat jika masalah muncul. Simpan logika asli sebagai cadangan terverifikasi, dan pertahankan prosedur rollback terperinci yang menentukan pemicu, tanggung jawab, dan langkah pemulihan.

Uji skenario rollback ini secara teratur dalam berbagai kondisi untuk memvalidasi keefektifannya, dan pastikan bahwa tim terbiasa dengan prosedur darurat. Bendera fitur juga memungkinkan peluncuran bertahap dengan mengaktifkan fungsionalitas baru secara selektif untuk grup atau transaksi pengguna tertentu. Ini memberikan lapisan tambahan mitigasi risiko selama transisi.

Memisahkan hubungan tabel selama dekomposisi database

Bagian ini memberikan panduan untuk memecah hubungan tabel yang kompleks dan operasi JOIN selama dekomposisi basis data monolitik. Gabungan tabel menggabungkan baris dari dua atau lebih tabel berdasarkan kolom terkait di antara mereka. Tujuan memisahkan hubungan ini adalah untuk mengurangi kopling tinggi antar tabel sambil mempertahankan integritas data di seluruh layanan mikro.

Bagian ini berisi topik berikut:

- Strategi denormalisasi
- Reference-by-key strategi
- Pola CQRS
- Sinkronisasi data berbasis peristiwa
- Menerapkan alternatif untuk bergabung tabel
- Contoh berbasis skenario

Strategi denormalisasi

Denormalisasi adalah strategi desain database yang melibatkan sengaja memperkenalkan redundansi dengan menggabungkan atau menduplikasi data di seluruh tabel. Ketika memecah database besar menjadi database kecil, mungkin masuk akal untuk menduplikasi beberapa data di seluruh layanan. Misalnya, menyimpan detail pelanggan dasar, seperti nama dan alamat email, baik dalam layanan pemasaran maupun layanan pesanan menghilangkan kebutuhan akan pencarian lintas layanan yang konstan. Layanan pemasaran mungkin memerlukan preferensi pelanggan dan informasi kontak untuk penargetan kampanye, sementara layanan pesanan memerlukan data yang sama untuk pemrosesan pesanan dan pemberitahuan. Meskipun ini menciptakan beberapa redundansi data, ini dapat secara signifikan meningkatkan kinerja dan kemandirian layanan, memungkinkan tim pemasaran untuk mengoperasikan kampanye mereka tanpa bergantung pada pencarian layanan pelanggan waktu nyata.

Saat menerapkan denormalisasi, fokuslah pada bidang yang sering diakses yang Anda identifikasi melalui analisis pola akses data yang cermat. Anda dapat menggunakan alat, Oracle AWR laporan semacam itu ataupg_stat_statements, untuk memahami data mana yang biasanya diambil bersama. Pakar domain juga dapat memberikan wawasan berharga tentang pengelompokan data alami.

Ingatlah bahwa denormalisasi bukanlah all-or-nothing pendekatan—hanya data duplikat yang terbukti meningkatkan kinerja sistem atau mengurangi dependensi kompleks.

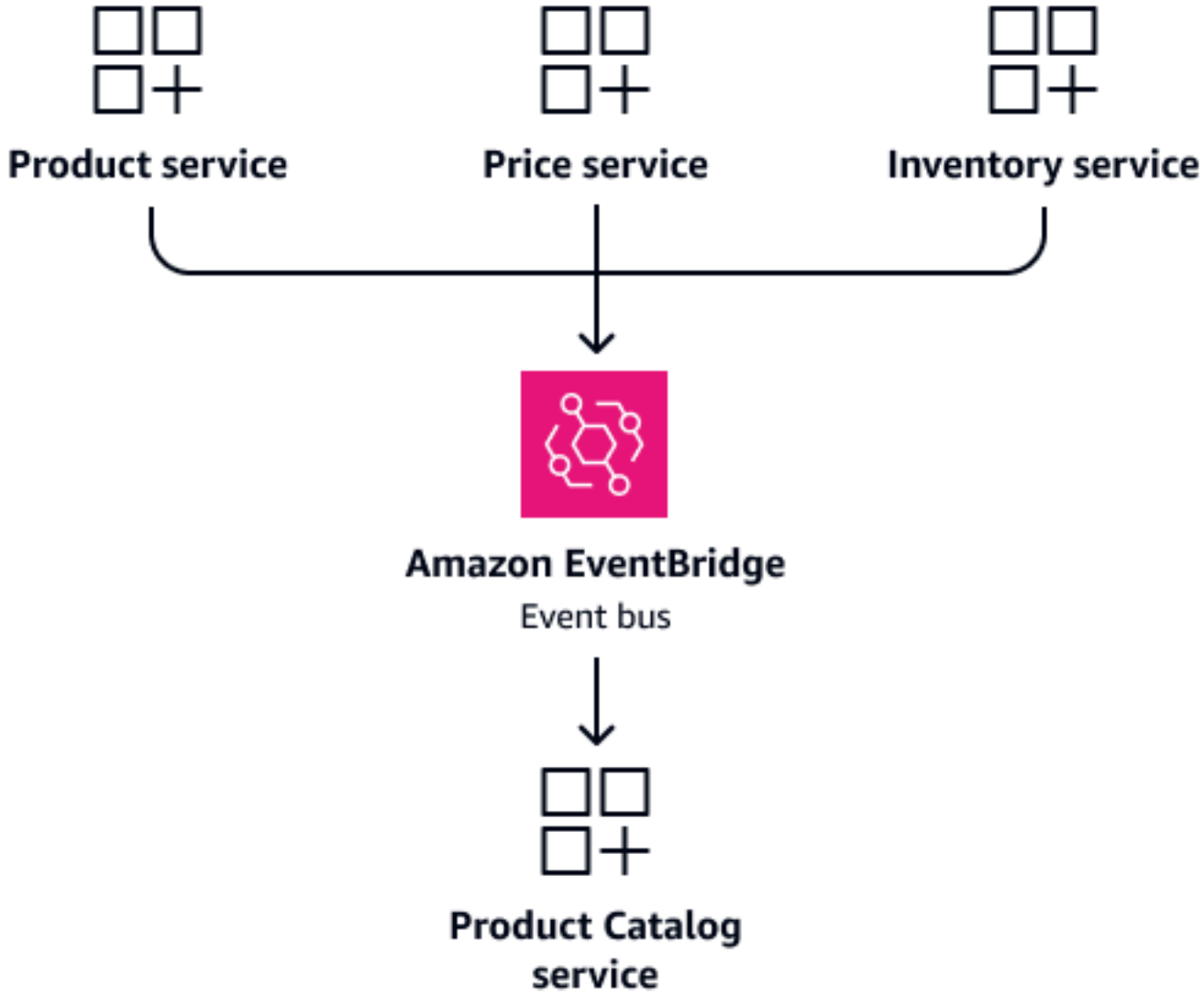
Reference-by-key strategi

reference-by-key Strategi adalah pola desain database di mana hubungan antar entitas dipertahankan melalui kunci unik daripada menyimpan data terkait yang sebenarnya. Alih-alih hubungan kunci asing tradisional, layanan mikro modern sering hanya menyimpan pengidentifikasi unik dari data terkait. Misalnya, daripada menyimpan semua detail pelanggan di tabel pesanan, layanan pesanan hanya menyimpan ID pelanggan dan mengambil informasi pelanggan tambahan melalui panggilan API bila diperlukan. Pendekatan ini menjaga independensi layanan sambil memastikan akses ke data terkait.

Pola CQRS

Pola Command Query Responsibility Segregation (CQRS) memisahkan operasi baca dan tulis dari penyimpanan data. Pola ini sangat berguna dalam sistem yang kompleks dengan persyaratan kinerja tinggi, terutama yang memiliki beban asimetris read/write . Jika aplikasi Anda sering membutuhkan data yang digabungkan dari berbagai sumber, Anda dapat membuat model CQRS khusus alih-alih gabungan yang kompleks. Misalnya, daripada bergabung`Product`,`Pricing`, dan `Inventory` tabel pada setiap permintaan, pertahankan `Product Catalog` tabel konsolidasi yang berisi data yang diperlukan. Manfaat dari pendekatan ini bisa lebih besar daripada biaya tabel tambahan.

Pertimbangkan skenario di mana`Product`,`Price`, dan `Inventory` layanan sering membutuhkan informasi produk. Alih-alih mengonfigurasi layanan ini untuk mengakses tabel bersama secara langsung, buat `Product Catalog` layanan khusus. Layanan ini memelihara basis datanya sendiri yang berisi informasi produk terkonsolidasi. Ini bertindak sebagai sumber kebenaran tunggal untuk pertanyaan terkait produk. Ketika detail produk, harga, atau tingkat inventaris berubah, masing-masing layanan dapat mempublikasikan acara untuk memperbarui `Product Catalog` layanan. Ini memberikan konsistensi data sambil mempertahankan independensi layanan. Gambar berikut menunjukkan konfigurasi ini, di mana Amazon EventBridge berfungsi sebagai bus acara.



Seperti yang dibahas diSinkronisasi data berbasis peristiwa, bagian selanjutnya, perbarui model CQRS melalui acara. Ketika detail produk, harga, atau tingkat inventaris berubah, masing-masing layanan mempublikasikan acara. `Product Catalog`Layanan berlangganan acara ini dan memperbarui tampilan konsolidasinya. Ini memberikan pembacaan cepat tanpa gabungan yang rumit, dan mempertahankan kemandirian layanan.

Sinkronisasi data berbasis peristiwa

Sinkronisasi data berbasis peristiwa adalah pola di mana perubahan data ditangkap dan disebarkan sebagai peristiwa, yang memungkinkan sistem atau komponen yang berbeda untuk mempertahankan status data yang disinkronkan. Saat data berubah, alih-alih segera memperbarui semua database terkait, publikasikan acara untuk memberi tahu layanan berlangganan.

Misalnya, ketika pelanggan mengubah alamat pengiriman mereka di `Customer` layanan, sebuah `CustomerUpdated` acara memulai pembaruan ke `Order` layanan dan `Delivery` layanan pada jadwal setiap layanan. Pendekatan ini menggantikan gabungan tabel kaku dengan pembaruan berbasis peristiwa yang fleksibel dan dapat diskalakan. Beberapa layanan mungkin secara singkat memiliki data yang sudah ketinggalan zaman, tetapi trade-off adalah peningkatan skalabilitas sistem dan independensi layanan.

Menerapkan alternatif untuk bergabung tabel

Mulailah dekomposisi database Anda dengan operasi baca karena biasanya lebih mudah untuk dimigrasi dan divalidasi. Setelah jalur baca stabil, atasi operasi penulisan yang lebih kompleks. Untuk persyaratan kritis dan kinerja tinggi, pertimbangkan untuk menerapkan pola CQRS. Gunakan database terpisah yang dioptimalkan untuk membaca sambil mempertahankan database lain untuk menulis.

Bangun sistem tangguh dengan menambahkan logika coba lagi untuk panggilan lintas layanan dan menerapkan lapisan caching yang sesuai. Pantau interaksi layanan dengan cermat, dan siapkan peringatan untuk masalah konsistensi data. Tujuan akhirnya bukanlah konsistensi sempurna di mana-mana—itu menciptakan layanan independen yang berkinerja baik sambil mempertahankan akurasi data yang dapat diterima untuk kebutuhan bisnis Anda.

Sifat layanan mikro yang dipisahkan memperkenalkan kompleksitas baru berikut dalam manajemen data:

- Data didistribusikan. Data sekarang berada di database terpisah, yang dikelola oleh layanan independen.
- Sinkronisasi real-time di seluruh layanan seringkali tidak praktis, memerlukan model konsistensi akhirnya.
- Operasi yang sebelumnya terjadi dalam satu transaksi database sekarang menjangkau beberapa layanan.

Untuk mengatasi tantangan ini, lakukan hal berikut:

- Menerapkan arsitektur berbasis peristiwa — Gunakan antrian pesan dan penerbitan acara untuk menyebarkan perubahan data di seluruh layanan. Untuk informasi selengkapnya, lihat Membangun Arsitektur Berbasis Acara di Tanah Tanpa Server.

- Mengadopsi pola orkestrasi saga — Pola ini membantu Anda mengelola transaksi terdistribusi dan menjaga integritas data di seluruh layanan. Untuk informasi selengkapnya, lihat Membangun aplikasi terdistribusi tanpa server menggunakan pola orkestrasi saga di Blog. AWS
- Desain untuk kegagalan — Menggabungkan mekanisme coba lagi, pemutus sirkuit, dan transaksi kompensasi untuk menangani masalah jaringan atau kegagalan layanan.
- Gunakan stamping versi — Lacak versi data untuk mengelola konflik dan pastikan pembaruan terbaru diterapkan.
- Regular rekonsiliasi — Menerapkan proses sinkronisasi data periodik untuk menangkap dan memperbaiki ketidakkonsistenan apa pun.

Contoh berbasis skenario

Contoh skema berikut memiliki dua tabel, `Customer` tabel dan `Order` tabel:

```
-- Customer table
CREATE TABLE customer (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    email VARCHAR(255),
    phone VARCHAR(20),
    address TEXT,
    created_at TIMESTAMP
);

-- Order table
CREATE TABLE order (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date TIMESTAMP,
    total_amount DECIMAL(10,2),
    status VARCHAR(50),
    FOREIGN KEY (customer_id) REFERENCES customers(id)
);
```

Berikut ini adalah contoh bagaimana Anda dapat menggunakan pendekatan denormalisasi:

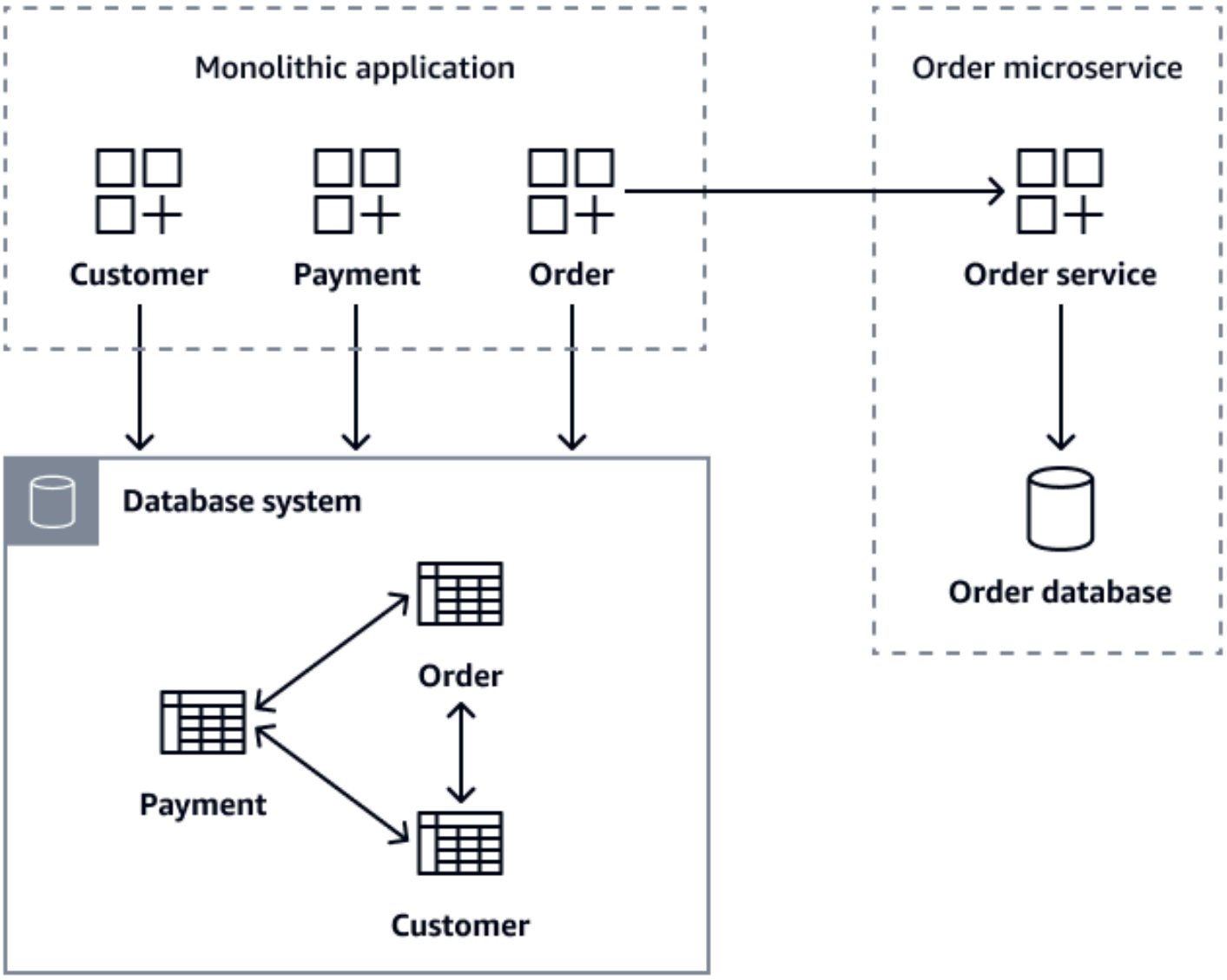
```
CREATE TABLE order (
    order_id INT PRIMARY KEY,
```

```
    customer_id INT,                  -- Reference only
    customer_first_name VARCHAR(100), -- Denormalized
    customer_last_name VARCHAR(100),  -- Denormalized
    customer_email VARCHAR(255),      -- Denormalized
    order_date TIMESTAMP,
    total_amount DECIMAL(10,2),
    status VARCHAR(50)
);
```

`Order`Tabel baru memiliki nama pelanggan dan alamat email yang didenormalisasi. `customer_id`Ini direferensikan, dan tidak ada kendala kunci asing dengan tabel. `Customer` Berikut ini adalah manfaat dari pendekatan denormalisasi ini:

- `Order`Layanan ini dapat menampilkan riwayat pesanan dengan detail pelanggan, dan tidak memerlukan panggilan API ke `Customer` layanan mikro.
- Jika `Customer` layanan sedang down, `Order` layanan tetap berfungsi penuh.
- Kueri untuk pemrosesan dan pelaporan pesanan berjalan lebih cepat.

Diagram berikut menunjukkan aplikasi monolitik yang mengambil data pesanan menggunakan`getOrder(customer_id)`, `getOrder(order_id)``getCustomerOders(customer_id)`, dan panggilan `createOrder(Order order)` API ke layanan mikro. `Order`



Selama migrasi layanan mikro, Anda dapat mempertahankan `Order` tabel dalam database monolitik sebagai tindakan keamanan transisi, memastikan bahwa aplikasi lama tetap berfungsi. Namun, sangat penting bahwa semua operasi terkait pesanan baru dirutekan melalui API `Order` layanan mikro, yang memelihara basis datanya sendiri sambil secara bersamaan menulis ke database lama sebagai cadangan. Pola penulisan ganda ini menyediakan jaring pengaman. Hal ini memungkinkan untuk migrasi bertahap sambil menjaga stabilitas sistem. Setelah semua pelanggan berhasil bermigrasi ke layanan mikro baru, Anda dapat menghentikan tabel lama `Order` di database monolitik. Setelah menguraikan aplikasi monolitik dan database-nya menjadi `Order` layanan terpisah `Customer` dan mikro, menjaga konsistensi data menjadi tantangan utama.

Praktik terbaik untuk dekomposisi basis data

Saat menguraikan basis data monolitik, organisasi harus menetapkan kerangka kerja yang jelas untuk melacak kemajuan, mempertahankan pengetahuan sistem, dan mengatasi tantangan yang muncul. Bagian ini memberikan praktik terbaik untuk mengukur keberhasilan dekomposisi, memelihara dokumentasi penting, menerapkan proses perbaikan berkelanjutan, dan menavigasi tantangan umum. Memahami dan mengikuti pedoman ini membantu Anda memastikan bahwa upaya dekomposisi basis data memberikan manfaat yang diinginkan sambil meminimalkan gangguan operasional dan utang teknis.

Bagian ini berisi topik berikut:

- Mengukur kesuksesan
- Persyaratan dokumentasi
- Strategi perbaikan berkelanjutan
- Mengatasi tantangan umum dalam dekomposisi basis data

Mengukur kesuksesan

Lacak keberhasilan dekomposisi melalui campuran metrik teknis, operasional, dan bisnis. Secara teknis, pantau waktu respons kueri, peningkatan uptime sistem, dan frekuensi penerapan meningkat. Secara operasional, mengukur pengurangan insiden, kecepatan resolusi masalah, dan peningkatan pemanfaatan sumber daya. Untuk pengembangan, lacak kecepatan implementasi fitur, akselerasi siklus rilis, dan pengurangan dependensi lintas tim. Dampak bisnis harus menghasilkan pengurangan biaya operasional, lebih cepat time-to-market, dan peningkatan kepuasan pelanggan. Metrik ini sering didefinisikan selama fase lingkup. Untuk informasi selengkapnya, lihat Mendefinisikan ruang lingkup dan persyaratan untuk dekomposisi basis data dalam panduan ini.

Persyaratan dokumentasi

Pertahankan dokumentasi arsitektur up-to-date sistem dengan batasan layanan yang jelas, aliran data, dan spesifikasi antarmuka. Gunakan catatan keputusan arsitektur (ADRs) untuk menangkap keputusan teknis utama, termasuk konteksnya, konsekuensi, dan alternatif yang dipertimbangkan. Misalnya, dokumentasikan mengapa layanan tertentu dipisahkan terlebih dahulu atau bagaimana pertukaran konsistensi data tertentu dibuat.

Jadwalkan tinjauan arsitektur bulanan untuk menilai kesehatan sistem melalui metrik utama: tren kinerja, kepatuhan keamanan, dan dependensi lintas layanan. Sertakan umpan balik dari tim pengembangan tentang tantangan integrasi dan masalah operasional. Siklus tinjauan reguler ini membantu Anda mengidentifikasi masalah yang muncul lebih awal dan memvalidasi bahwa upaya dekomposisi tetap selaras dengan tujuan bisnis.

Strategi perbaikan berkelanjutan

Perlakukan dekomposisi database sebagai proses berulang, bukan proyek satu kali. Pantau metrik kinerja sistem dan interaksi layanan untuk mengidentifikasi peluang pengoptimalan. Setiap kuartal, memprioritaskan penanganan utang teknis berdasarkan dampak operasional dan biaya pemeliharaan. Misalnya, mengotomatiskan operasi database yang sering dilakukan, meningkatkan cakupan pemantauan, dan menyempurnakan prosedur penerapan berdasarkan pola yang dipelajari.

Mengatasi tantangan umum dalam dekomposisi basis data

Optimalisasi kinerja membutuhkan pendekatan multi-segi. Menerapkan caching strategis pada batas layanan, mengoptimalkan pola kueri berdasarkan penggunaan aktual, dan terus memantau metrik kunci. Atasi kemacetan kinerja secara proaktif dengan menganalisis tren dan menetapkan ambang batas yang jelas untuk intervensi.

Tantangan konsistensi data menuntut pilihan arsitektur yang cermat. Menerapkan pola berbasis peristiwa untuk pembaruan lintas layanan dan gunakan pola orkestrasi saga untuk transaksi yang kompleks. Tentukan batasan layanan yang jelas, dan terima konsistensi akhirnya di mana persyaratan bisnis mengizinkan. Keseimbangan antara konsistensi dan otonomi layanan ini sangat penting untuk keberhasilan dekomposisi.

Keunggulan operasional membutuhkan otomatisasi tugas rutin dan prosedur standar di seluruh layanan. Pertahankan pemantauan komprehensif dengan ambang peringatan yang jelas, dan investasikan dalam pelatihan tim reguler untuk pola dan alat baru. Pendekatan sistematis untuk operasi ini mempromosikan penyampaian layanan yang andal sambil mengelola kompleksitas.

FAQ untuk dekomposisi basis data

Bagian FAQ komprehensif ini membahas pertanyaan dan tantangan paling umum yang dihadapi organisasi saat melakukan proyek dekomposisi basis data. Dari mendefinisikan ruang lingkup awal dan persyaratan untuk memigrasikan prosedur tersimpan, pertanyaan-pertanyaan ini memberikan wawasan praktis dan pendekatan strategis untuk membantu tim berhasil menavigasi perjalanan modernisasi database mereka. Apakah Anda sedang dalam tahap perencanaan atau sudah menjalankan strategi dekomposisi Anda, jawaban ini dapat membantu Anda menghindari jebakan umum dan menerapkan praktik terbaik untuk hasil yang optimal.

Bagian ini berisi topik berikut:

FAQs tentang mendefinisikan ruang lingkup dan persyaratan

Mendefinisikan ruang lingkup dan persyaratan untuk dekomposisi databaseBagian dari panduan ini membahas bagaimana menganalisis interaksi, memetakan dependensi, dan menetapkan kriteria keberhasilan. Bagian FAQ ini membahas pertanyaan kunci tentang menetapkan dan mengelola batas-batas proyek. Apakah Anda berurusan dengan kendala teknis yang tidak jelas, kebutuhan departemen yang bertentangan, atau persyaratan bisnis yang berkembang, ini FAQs memberikan panduan praktis untuk mempertahankan pendekatan yang seimbang.

Bagian ini berisi pertanyaan-pertanyaan berikut:

- Bagaimana cara memastikan bahwa saya tidak melewatkan persyaratan penting dari pemangku kepentingan diam?
- Apakah rekomendasi ini berlaku untuk database mainframe monolitik?

Seberapa rinci definisi ruang lingkup awal?

Bekerja mundur dari kebutuhan pelanggan Anda, tentukan ruang lingkup proyek dengan detail yang cukup untuk mengidentifikasi batas-batas sistem dan dependensi kritis sambil mempertahankan fleksibilitas untuk penemuan. Memetakan elemen-elemen penting, termasuk antarmuka sistem, pemangku kepentingan utama, dan kendala teknis utama. Mulailah dari yang kecil dengan memilih bagian terbatas dan berisiko rendah dari sistem yang memberikan nilai terukur. Pendekatan ini membantu tim untuk belajar dan menyesuaikan strategi sebelum menangani komponen yang lebih kompleks.

Dokumentasikan persyaratan bisnis penting yang mendorong upaya dekomposisi, tetapi hindari perincian yang terlalu spesifik yang mungkin berubah selama implementasi. Pendekatan seimbang ini memastikan bahwa tim dapat bergerak maju dengan jelas sambil tetap beradaptasi dengan wawasan dan tantangan baru yang muncul selama perjalanan modernisasi.

Bagaimana jika saya menemukan dependensi tambahan setelah memulai proyek?

Berharap untuk mengungkap dependensi tambahan saat proyek berlangsung. Pertahankan log ketergantungan langsung dan lakukan tinjauan ruang lingkup reguler untuk menilai dampak pada jadwal dan sumber daya. Menerapkan proses manajemen perubahan yang jelas, dan sertakan waktu penyangga dalam rencana proyek untuk menangani penemuan yang tidak terduga. Tujuannya bukan untuk mencegah perubahan tetapi untuk mengelolanya secara efektif. Ini membantu tim untuk beradaptasi dengan cepat sambil mempertahankan momentum proyek.

Bagaimana cara menangani pemangku kepentingan dari berbagai departemen yang memiliki persyaratan yang bertentangan?

Menangani persyaratan departemen yang bertentangan melalui prioritas yang jelas yang didasarkan pada nilai bisnis dan dampak sistem. Mengamankan sponsor eksekutif untuk mendorong keputusan penting dan menyelesaikan konflik dengan cepat. Jadwalkan pertemuan penyelarasan pemangku kepentingan secara teratur untuk membahas trade-off dan menjaga transparansi.

Dokumentasikan semua keputusan dan alasannya untuk mempromosikan komunikasi yang jelas dan mempertahankan momentum proyek. Fokus diskusi pada manfaat bisnis yang dapat diukur daripada preferensi departemen.

Apa cara terbaik untuk menilai kendala teknis ketika dokumentasi buruk atau usang?

Saat menghadapi dokumentasi yang buruk, gabungkan analisis tradisional dengan alat AI modern. Gunakan model bahasa besar (LLMs) untuk menganalisis repositori kode, log, dan dokumentasi yang ada untuk mengidentifikasi pola dan kendala potensial. Wawancara pengembang berpengalaman dan arsitek database untuk memvalidasi temuan AI dan mengungkap kendala yang tidak berdokumen. Terapkan alat pemantauan yang telah meningkatkan kemampuan AI untuk mengamati perilaku sistem dan memprediksi potensi masalah.

Buat eksperimen teknis kecil yang memvalidasi asumsi Anda. Anda dapat menggunakan alat pengujian bertenaga AI untuk mempercepat proses. Dokumentasikan temuan dalam basis pengetahuan yang dapat terus ditingkatkan melalui pembaruan yang dibantu AI. Pertimbangkan untuk melibatkan ahli materi pelajaran untuk area yang kompleks, dan gunakan alat pemrograman pasangan AI untuk mempercepat upaya analisis dan dokumentasi mereka.

Bagaimana cara menyeimbangkan kebutuhan bisnis langsung dengan tujuan teknis jangka panjang?

Buat peta jalan proyek bertahap yang menyelaraskan kebutuhan bisnis langsung dengan tujuan teknis jangka panjang. Identifikasi kemenangan cepat yang memberikan nilai nyata lebih awal sehingga Anda dapat membangun kepercayaan pemangku kepentingan. Memecah dekomposisi menjadi tonggak yang jelas. Masing-masing harus memberikan manfaat bisnis yang terukur sambil maju menuju tujuan arsitektur. Pertahankan fleksibilitas untuk memenuhi kebutuhan bisnis yang mendesak melalui tinjauan dan penyesuaian peta jalan reguler.

Bagaimana cara memastikan bahwa saya tidak melewatkan persyaratan penting dari pemangku kepentingan diam?

Memetakan semua pemangku kepentingan potensial di seluruh organisasi, termasuk pemilik sistem hilir dan pengguna tidak langsung. Buat beberapa saluran umpan balik melalui wawancara terstruktur, lokakarya, dan sesi tinjauan reguler. Membangun proof-of-concepts dan membuat prototipe untuk membuat persyaratan nyata dan memicu diskusi yang bermakna. Misalnya,

dasbor sederhana yang menunjukkan ketergantungan sistem sering mengungkapkan pemangku kepentingan tersembunyi dan persyaratan yang awalnya tidak terlihat.

Lakukan sesi validasi reguler dengan pemangku kepentingan yang vokal dan tenang, dan pastikan bahwa semua perspektif ditangkap. Wawasan kritis sering datang dari mereka yang paling dekat dengan operasi sehari-hari daripada suara paling keras dalam pertemuan perencanaan.

Apakah rekomendasi ini berlaku untuk database mainframe monolitik?

Metodologi yang dijelaskan dalam panduan ini juga berlaku untuk menguraikan basis data mainframe monolitik. Tantangan utama dengan database ini adalah mengelola persyaratan dari berbagai pemangku kepentingan. Rekomendasi teknologi dalam panduan ini mungkin berlaku untuk database mainframe monolitik. Jika mainframe memiliki database relasional, seperti database pemrosesan transaksi online (OLTP), maka banyak rekomendasi yang berlaku. Untuk database pemrosesan analitik online (OLAP), seperti yang digunakan untuk menghasilkan laporan bisnis, maka hanya beberapa rekomendasi yang berlaku.

FAQs tentang mengendalikan akses database

Mengontrol akses database dengan menggunakan pola layanan pembungkus database dibahas di Mengontrol akses database selama dekomposisi bagian panduan ini. Bagian FAQ ini membahas masalah dan pertanyaan umum tentang memperkenalkan layanan pembungkus basis data, termasuk potensi dampaknya terhadap kinerja, penanganan prosedur tersimpan yang ada, mengelola transaksi kompleks, dan mengawasi perubahan skema.

Bagian ini berisi pertanyaan-pertanyaan berikut:

- Bukankah layanan pembungkus akan menjadi hambatan baru?
- Apa yang terjadi pada prosedur tersimpan yang ada?
- Bagaimana cara mengelola perubahan skema selama transisi?

Bukankah layanan pembungkus akan menjadi hambatan baru?

Sementara layanan pembungkus database memang menambahkan hop jaringan tambahan, dampaknya biasanya minimal. Anda dapat menskalakan layanan secara horizontal, dan manfaat akses terkontrol biasanya lebih besar daripada biaya kinerja yang kecil. Anggap saja sebagai trade-off sementara antara kinerja dan pemeliharaan.

Apa yang terjadi pada prosedur tersimpan yang ada?

Awalnya, layanan pembungkus database dapat mengekspos prosedur tersimpan sebagai metode layanan. Seiring waktu, Anda dapat secara bertahap memindahkan logika ke lapisan aplikasi, yang meningkatkan pengujian dan kontrol versi. Migrasikan logika bisnis secara bertahap untuk meminimalkan risiko.

Bagaimana cara mengelola perubahan skema selama transisi?

Memusatkan kontrol perubahan skema melalui tim layanan pembungkus. Tim ini bertanggung jawab untuk menjaga visibilitas komprehensif di semua konsumen. Tim ini meninjau perubahan yang diusulkan untuk dampak seluruh sistem, berkoordinasi dengan tim yang terpengaruh, dan mengimplementasikan modifikasi dengan menggunakan proses penyebaran terkontrol. Misalnya, saat menambahkan bidang baru, tim ini harus mempertahankan kompatibilitas mundur dengan menerapkan nilai default atau awalnya mengizinkan nol.

Menetapkan proses manajemen perubahan yang jelas yang mencakup penilaian dampak, persyaratan pengujian, dan prosedur rollback. Gunakan alat pembuatan versi database, dan pertahankan dokumentasi yang jelas dari semua perubahan. Pendekatan terpusat ini mencegah modifikasi skema mengganggu layanan dependen dan menjaga stabilitas sistem.

FAQs tentang menganalisis kohesi dan kopling

Memahami dan menganalisis kopling dan kohesi basis data secara efektif merupakan dasar untuk keberhasilan dekomposisi basis data. Kopling dan kohesi dibahas di Menganalisis kohesi dan kopling untuk dekomposisi basis data bagian panduan ini. Bagian FAQ ini membahas pertanyaan kunci tentang mengidentifikasi tingkat perincian yang sesuai, memilih alat analisis yang tepat, mendokumentasikan temuan, dan memprioritaskan masalah kopling.

Bagian ini berisi pertanyaan-pertanyaan berikut:

- Bagaimana cara mengidentifikasi tingkat granularitas yang tepat saat menganalisis kopling?
- Alat apa yang dapat saya gunakan untuk menganalisis kopling dan kohesi basis data?
- Apa cara terbaik untuk mendokumentasikan temuan kopling dan kohesi?
- Bagaimana cara memprioritaskan masalah kopling mana yang harus diatasi terlebih dahulu?
- Bagaimana cara menangani transaksi yang mencakup beberapa operasi?

Bagaimana cara mengidentifikasi tingkat granularitas yang tepat saat menganalisis kopling?

Mulailah dengan analisis luas hubungan database, kemudian secara sistematis menelusuri untuk mengidentifikasi titik pemisahan alami. Gunakan alat analisis database untuk memetakan hubungan tingkat tabel, dependensi skema, dan batas transaksi. Misalnya, periksa pola gabungan dalam kueri SQL untuk memahami dependensi akses data. Anda juga dapat menganalisis log transaksi untuk mengidentifikasi batas-batas proses bisnis.

Fokus pada area di mana kopling secara alami minimal. Ini sering selaras dengan batas domain bisnis dan mewakili titik dekomposisi yang optimal. Saat menentukan batas layanan yang sesuai, pertimbangkan kopling teknis (seperti tabel bersama dan kunci asing) dan kopling bisnis (seperti arus proses dan kebutuhan pelaporan).

Alat apa yang dapat saya gunakan untuk menganalisis kopling dan kohesi basis data?

Anda dapat menggunakan kombinasi alat otomatis dan analisis manual untuk menilai kopling dan kohesi basis data. Alat-alat berikut dapat membantu Anda dengan penilaian ini:

- Alat visualisasi skema — Anda dapat menggunakan alat seperti SchemaSpyatau pgAdminuntuk menghasilkan diagram ER. Diagram ini mengungkapkan hubungan tabel dan titik kopling potensial.
- Alat analisis kueri — Anda dapat menggunakan pg_stat_statementsatau SQL Server Query Storemengidentifikasi tabel dan pola akses yang sering digabungkan.
- Alat pembuatan profil basis data — Alat seperti Oracle SQL Developeratau MySQL Workbenchmemberikan wawasan tentang kinerja kueri dan dependensi data.
- Alat pemetaan ketergantungan — The AWS Schema Conversion Tool (AWS SCT) dapat membantu Anda memvisualisasikan hubungan skema dan mengidentifikasi komponen yang digabungkan secara erat. vFunctiondapat membantu Anda mengidentifikasi batas domain dengan menganalisis batas fungsional dan domain aplikasi.
- Alat pemantauan transaksi — Anda dapat menggunakan alat khusus database, seperti Oracle Enterprise Manageratau SQL Server Extended Events, untuk menganalisis batas-batas transaksi.
- Alat migrasi logika bisnis — Anda dapat menggunakan Ispireratau alat AI generatif, seperti Amazon Q Developer atau Kiro, untuk mengonversi logika bisnis basis data untuk lapisan aplikasi, seperti konversi ke Java.

Gabungkan analisis otomatis ini dengan tinjauan manual proses bisnis dan pengetahuan domain untuk sepenuhnya memahami kopling sistem. Pendekatan multi-segi ini memastikan bahwa perspektif teknis dan bisnis dipertimbangkan dalam strategi dekomposisi Anda.

Apa cara terbaik untuk mendokumentasikan temuan kopling dan kohesi?

Buat dokumentasi komprehensif yang memvisualisasikan hubungan database dan pola penggunaan. Berikut ini adalah jenis aset yang dapat Anda gunakan untuk mencatat temuan Anda:

- Matriks ketergantungan — Petakan dependensi tabel dan sorot area kopling tinggi.
- Diagram hubungan — Gunakan diagram ER untuk menunjukkan koneksi skema dan hubungan kunci asing.
- Peta panas penggunaan tabel — Visualisasikan frekuensi kueri dan pola akses data di seluruh tabel.
- Diagram alur transaksi — Dokumentasikan transaksi multi-tabel dan batas-batasnya.
- Peta batas domain — Garis besar batas layanan potensial berdasarkan domain bisnis.

Gabungkan artefak ini dalam dokumen, dan perbarui secara teratur saat dekomposisi berlangsung. Untuk diagram, Anda dapat menggunakan alat seperti draw.ioatau. Lucidchart Pertimbangkan untuk menerapkan wiki untuk memudahkan akses dan kolaborasi tim. Pendekatan dokumentasi multi-segi ini memberikan pemahaman yang jelas dan bersama tentang kopling dan kohesi sistem.

Bagaimana cara memprioritaskan masalah kopling mana yang harus diatasi terlebih dahulu?

Memprioritaskan masalah kopling berdasarkan penilaian yang seimbang dari faktor bisnis dan teknis. Mengevaluasi setiap masalah terhadap dampak bisnis (seperti pendapatan dan pengalaman pelanggan), risiko teknis (seperti stabilitas sistem dan integritas data), upaya implementasi, dan kemampuan tim. Buat matriks prioritas yang menilai setiap masalah dari 1-5 di seluruh dimensi ini. Matriks ini membantu Anda mengidentifikasi peluang paling berharga dengan risiko yang dapat dikelola.

Mulailah dengan perubahan berdampak tinggi dan berisiko rendah yang selaras dengan keahlian tim yang ada. Ini membantu Anda membangun kepercayaan dan momentum organisasi untuk perubahan yang lebih kompleks. Pendekatan ini mempromosikan eksekusi yang realistis dan memaksimalkan nilai bisnis. Secara teratur meninjau dan menyesuaikan prioritas untuk membantu menjaga keselarasan dengan perubahan kebutuhan bisnis dan kapasitas tim.

Bagaimana cara menangani transaksi yang mencakup beberapa operasi?

Menangani transaksi multi-operasi melalui koordinasi tingkat layanan yang dirancang dengan cermat. Menerapkan pola saga untuk transaksi terdistribusi yang kompleks. Pecah menjadi langkah-langkah yang lebih kecil dan dapat dibalik yang dapat dikelola secara independen. Misalnya, alur pemrosesan pesanan dapat dibagi menjadi langkah-langkah terpisah untuk pemeriksaan inventaris, pemrosesan pembayaran, dan pembuatan pesanan, masing-masing dengan mekanisme kompensasinya sendiri.

Jika memungkinkan, desain ulang operasi menjadi lebih atomik, yang mengurangi kebutuhan akan transaksi terdistribusi. Ketika transaksi terdistribusi tidak dapat dihindari, terapkan mekanisme pelacakan dan kompensasi yang kuat untuk mempromosikan konsistensi data. Pantau tingkat penyelesaian transaksi dan terapkan prosedur pemulihan kesalahan yang jelas untuk menjaga keandalan sistem.

FAQs tentang memigrasikan logika bisnis ke lapisan aplikasi

Migrasi logika bisnis dari database ke lapisan aplikasi adalah aspek penting dan kompleks dari modernisasi database. Migrasi logika bisnis ini dibahas di Migrasi logika bisnis dari database ke lapisan aplikasi bagian panduan ini. Bagian FAQ ini membahas pertanyaan umum tentang mengelola transisi ini secara efektif, mulai dari memilih kandidat awal untuk migrasi hingga menangani prosedur dan pemicu tersimpan yang kompleks.

Bagian ini berisi pertanyaan-pertanyaan berikut:

- Bagaimana cara mengidentifikasi prosedur tersimpan mana yang harus dimigrasikan terlebih dahulu?
- Apa risiko memindahkan logika ke lapisan aplikasi?
- Bagaimana cara mempertahankan kinerja saat memindahkan logika dari database?
- Apa yang harus saya lakukan dengan prosedur tersimpan kompleks yang melibatkan banyak tabel?
- Bagaimana cara menangani pemicu database selama migrasi?
- Apa cara terbaik untuk menguji logika bisnis yang dimigrasi?
- Bagaimana cara mengelola periode transisi ketika database dan logika aplikasi ada?
- Bagaimana cara menangani skenario kesalahan di lapisan aplikasi yang sebelumnya dikelola oleh database?

Bagaimana cara mengidentifikasi prosedur tersimpan mana yang harus dimigrasikan terlebih dahulu?

Mulailah dengan mengidentifikasi prosedur tersimpan yang menawarkan kombinasi terbaik dari nilai berisiko rendah dan pembelajaran tinggi. Fokus pada prosedur yang memiliki ketergantungan minimal, fungsionalitas yang jelas, dan dampak bisnis yang tidak kritis. Ini membuat kandidat ideal untuk migrasi awal karena mereka membantu tim membangun kepercayaan diri dan membangun pola. Misalnya, pilih prosedur yang menangani operasi data sederhana daripada yang mengelola transaksi kompleks atau logika bisnis kritis.

Gunakan alat pemantauan basis data untuk menganalisis pola penggunaan dan mengidentifikasi prosedur yang jarang diakses sebagai kandidat awal. Pendekatan ini meminimalkan risiko bisnis sambil memberikan pengalaman berharga untuk menangani migrasi yang lebih kompleks nantinya. Skor setiap prosedur pada tingkat kompleksitas, kekritisan bisnis, dan ketergantungan untuk membuat urutan migrasi yang diprioritaskan.

Apa risiko memindahkan logika ke lapisan aplikasi?

Memindahkan logika database ke lapisan aplikasi memperkenalkan beberapa tantangan utama. Kinerja sistem dapat menurun karena peningkatan panggilan jaringan, terutama untuk operasi intensif data yang sebelumnya ditangani dalam database. Manajemen transaksi menjadi lebih kompleks dan membutuhkan koordinasi yang cermat untuk menjaga integritas data di seluruh operasi terdistribusi. Memastikan konsistensi data menjadi tantangan, terutama untuk operasi yang sebelumnya mengandalkan kendala tingkat database.

Potensi gangguan bisnis selama migrasi dan kurva pembelajaran bagi pengembang juga menjadi perhatian yang signifikan. Mengurangi risiko ini melalui perencanaan menyeluruh, pengujian ekstensif di lingkungan bertahap, dan migrasi bertahap yang dimulai dengan komponen yang kurang kritis. Menerapkan prosedur pemantauan dan rollback yang kuat untuk mengidentifikasi dan mengatasi masalah dalam produksi dengan cepat.

Bagaimana cara mempertahankan kinerja saat memindahkan logika dari database?

Menerapkan mekanisme caching yang sesuai untuk data yang sering diakses, mengoptimalkan pola akses data untuk meminimalkan panggilan jaringan, dan menggunakan pemrosesan batch untuk operasi massal. Untuk non-time-critical operasi, pertimbangkan pemrosesan asinkron untuk meningkatkan respons sistem.

Pantau metrik kinerja aplikasi dengan cermat dan sesuaikan sesuai kebutuhan. Misalnya, Anda dapat mengganti beberapa operasi baris tunggal dengan pemrosesan massal, Anda dapat menyimpan data referensi cache yang jarang berubah, dan Anda dapat mengoptimalkan pola kueri untuk mengurangi transfer data. Pengujian dan penyetelan kinerja reguler membantu sistem mempertahankan waktu respons yang dapat diterima dan meningkatkan pemeliharaan dan skalabilitas.

Apa yang harus saya lakukan dengan prosedur tersimpan kompleks yang melibatkan banyak tabel?

Dekati prosedur tersimpan multi-tabel yang kompleks melalui dekomposisi sistematis. Mulailah dengan memecahnya menjadi komponen yang lebih kecil dan koheren secara logis, dan identifikasi batas transaksi dan dependensi data yang jelas. Buat antarmuka layanan untuk setiap komponen logis. Ini membantu Anda bermigrasi secara bertahap tanpa mengganggu fungsionalitas yang ada.

Menerapkan step-by-step migrasi, dimulai dengan komponen yang paling sedikit digabungkan. Untuk prosedur yang sangat rumit, pertimbangkan untuk menyimpannya sementara di database sambil memigrasikan bagian yang lebih sederhana. Pendekatan hibrida ini menjaga stabilitas sistem saat Anda maju menuju tujuan arsitektur Anda. Terus memantau kinerja dan fungsionalitas selama migrasi, dan bersiaplah untuk menyesuaikan strategi Anda berdasarkan hasil.

Bagaimana cara menangani pemicu database selama migrasi?

Ubah pemicu database menjadi event handler tingkat aplikasi sambil mempertahankan fungsionalitas sistem. Ganti pemicu sinkron dengan pola berbasis peristiwa yang mengantrian pesan untuk operasi asinkron. Pertimbangkan untuk menggunakan Amazon Simple Notification Service (Amazon SNS) atau Amazon Simple Queue Service (Amazon SQS) untuk antrian pesan. Untuk persyaratan audit, terapkan pencatatan tingkat aplikasi atau gunakan fitur pengambilan data perubahan basis data (CDC).

Analisis tujuan dan kekritisan setiap pemicu. Beberapa pemicu mungkin lebih baik dilayani oleh logika aplikasi, dan yang lain mungkin memerlukan pola sumber peristiwa untuk menjaga konsistensi data. Mulailah dengan pemicu sederhana, seperti log audit, sebelum menangani pemicu kompleks yang mengelola aturan bisnis atau integritas data. Pantau dengan cermat selama migrasi untuk memastikan bahwa tidak ada kehilangan fungsionalitas atau konsistensi data.

Apa cara terbaik untuk menguji logika bisnis yang dimigrasi?

Terapkan pendekatan pengujian berlapis-lapis sebelum Anda menerapkan logika bisnis yang dimigrasi. Mulailah dengan pengujian unit untuk kode aplikasi baru, lalu tambahkan tes integrasi

yang mencakup alur end-to-end bisnis. Jalankan implementasi lama dan baru secara paralel, lalu bandingkan hasilnya untuk memvalidasi kesetaraan fungsional. Lakukan pengujian kinerja dalam berbagai kondisi beban untuk memverifikasi bahwa perilaku sistem cocok atau melebihi kemampuan sebelumnya.

Gunakan flag fitur untuk mengontrol penerapan sehingga Anda dapat dengan cepat memutar kembali jika masalah muncul. Libatkan pengguna bisnis dalam validasi, terutama untuk alur kerja kritis. Pantau metrik kunci selama penerapan awal, dan secara bertahap tingkatkan lalu lintas ke implementasi baru. Sepanjang, pertahankan kemampuan untuk kembali ke logika database asli jika diperlukan.

Bagaimana cara mengelola periode transisi ketika database dan logika aplikasi ada?

Ketika database dan logika aplikasi keduanya digunakan, terapkan flag fitur yang mengontrol arus lalu lintas dan memungkinkan peralihan cepat antara implementasi lama dan baru. Pertahankan kontrol versi yang ketat, dan dokumentasikan implementasi dan tanggung jawabnya dengan jelas. Siapkan pemantauan komprehensif untuk kedua sistem untuk dengan cepat mengidentifikasi perbedaan atau masalah kinerja.

Tetapkan prosedur rollback yang jelas untuk setiap komponen yang dimigrasi sehingga Anda dapat kembali ke logika asli jika diperlukan. Berkomunikasi secara teratur dengan semua pemangku kepentingan tentang status transisi, dampak potensial, dan prosedur eskalasi. Pendekatan ini membantu Anda bermigrasi secara bertahap sambil menjaga stabilitas sistem dan kepercayaan pemangku kepentingan.

Bagaimana cara menangani skenario kesalahan di lapisan aplikasi yang sebelumnya dikelola oleh database?

Ganti penanganan kesalahan tingkat database dengan mekanisme lapisan aplikasi yang kuat. Menerapkan pemutus sirkuit dan coba kembali logika untuk kegagalan sementara. Gunakan transaksi kompensasi untuk menjaga konsistensi data di seluruh operasi terdistribusi. Misalnya, jika pembaruan pembayaran gagal, aplikasi harus secara otomatis mencoba lagi dalam batas yang ditentukan dan memulai tindakan kompensasi jika diperlukan.

Siapkan pemantauan dan peringatan komprehensif untuk mengidentifikasi masalah dengan cepat, dan memelihara log audit terperinci untuk pemecahan masalah. Rancang penanganan kesalahan agar seotomatis mungkin, dan tentukan jalur eskalasi yang jelas untuk skenario yang memerlukan

intervensi manusia. Pendekatan berlapis-lapis ini memberikan ketahanan sistem sambil menjaga integritas data dan kelangsungan proses bisnis.

Langkah selanjutnya untuk dekomposisi basis data pada AWS

Setelah menerapkan strategi dekomposisi basis data awal melalui layanan pembungkus basis data dan memindahkan logika bisnis ke lapisan aplikasi, organisasi harus merencanakan evolusi berikutnya. Bagian ini menguraikan pertimbangan utama untuk melanjutkan perjalanan modernisasi Anda.

Bagian ini berisi topik berikut:

- Strategi tambahan untuk dekomposisi basis data
- Pertimbangan teknis untuk lingkungan database terdistribusi
- Perubahan organisasi untuk mendukung arsitektur terdistribusi

Strategi tambahan untuk dekomposisi basis data

Dekomposisi database mengikuti evolusi bertahap melalui tiga fase yang berbeda. Tim pertama membungkus database monolitik dengan layanan pembungkus database untuk mengontrol akses. Mereka kemudian mulai membagi data menjadi database khusus layanan, sambil mempertahankan basis data utama untuk kebutuhan warisan. Akhirnya, mereka menyelesaikan migrasi logika bisnis untuk transisi ke database layanan yang sepenuhnya independen.

Sepanjang perjalanan ini, tim harus menerapkan pola sinkronisasi data yang cermat dan terus memvalidasi konsistensi di seluruh layanan. Pemantauan kinerja menjadi penting untuk mengidentifikasi dan mengatasi masalah potensial sejak dini. Ketika layanan berkembang secara independen, skema mereka harus dioptimalkan berdasarkan pola penggunaan aktual, dan Anda harus menghapus struktur berlebihan yang terakumulasi dari waktu ke waktu.

Pendekatan inkremental ini membantu meminimalkan risiko sambil menjaga stabilitas sistem selama proses transformasi.

Pertimbangan teknis untuk lingkungan database terdistribusi

Dalam lingkungan database terdistribusi, pemantauan kinerja menjadi penting untuk mengidentifikasi dan mengatasi kemacetan sejak dini. Tim harus menerapkan sistem pemantauan yang komprehensif

dan strategi caching untuk mempertahankan tingkat kinerja. Read/write pemisahan dapat secara efektif menyeimbangkan beban di seluruh sistem.

Konsistensi data membutuhkan orkestrasi yang cermat di seluruh layanan terdistribusi. Tim harus menerapkan pola konsistensi akhirnya jika sesuai dan menetapkan batas kepemilikan data yang jelas. Pemantauan yang kuat meningkatkan integritas data di semua layanan.

Selain itu, keamanan harus berkembang untuk mengakomodasi arsitektur terdistribusi. Setiap layanan memerlukan kontrol keamanan yang halus, dan pola akses Anda memerlukan peninjauan rutin. Peningkatan pemantauan dan audit menjadi penting dalam lingkungan terdistribusi ini.

Perubahan organisasi untuk mendukung arsitektur terdistribusi

Struktur tim harus selaras dengan batasan layanan untuk menentukan kepemilikan dan akuntabilitas yang jelas. Organizations harus membangun pola komunikasi baru dan membangun kemampuan teknis tambahan dalam tim. Struktur ini harus mendukung pemeliharaan layanan yang ada dan evolusi arsitektur Anda yang berkelanjutan.

Anda harus memperbarui proses operasional Anda untuk menangani arsitektur terdistribusi. Tim harus memodifikasi prosedur penerapan, mengadaptasi proses respons insiden, dan mengembangkan praktik manajemen perubahan untuk berkoordinasi di berbagai layanan.

Sumber Daya

Sumber daya dan alat tambahan berikut dapat membantu organisasi Anda dalam perjalanan dekomposisi database-nya.

AWS Bimbingan Preskriptif

- Migrasi Oracle database ke AWS Cloud
- Opsi replatform untuk aktif Oracle DatabaseAWS
- Pola, arsitektur, dan implementasi desain cloud

AWS posting blog

- Migrasikan logika bisnis dari database ke aplikasi untuk inovasi dan fleksibilitas yang lebih cepat

Layanan AWS

- AWS Application Migration Service
- AWS Database Migration Service (AWS DMS)
- Migration Evaluator
- AWS Schema Conversion Tool (AWS SCT)
- AWS Transform

Alat-alat lainnya

- AppEngine(Dynatracesitus web)
- Oracle Automatic Workload Repository(Oraclesitus web)
- CAST Imaging(CASTsitus web)
- Kiro(Kirositus web)
- pgAdmin(pgAdminsitus web)
- pg_stat_statements(PostgreSQLsitus web)
- SchemaSpy(SchemaSpysitus web)

- SQL Developer(Oraclesitus web)
- SQLWays(Ispirersitus web)
- vFunction(vFunctionsitus web)

Sumber daya lainnya

- Monolith ke microservices (situs web) O'Reilly

Riwayat dokumen

Tabel berikut menjelaskan perubahan signifikan pada panduan ini. Jika Anda ingin diberi tahu tentang pembaruan masa depan, Anda dapat berlangganan umpan RSS.

Perubahan	Deskripsi	Tanggal
FAQ mainframe dan alat AI	Kami menambahkan Apakah rekomendasi ini berlaku untuk database mainframe monolitik? FAQ, dan kami menambahkan informasi tambahan tentang alat AI yang dapat Anda gunakan selama dekomposisi basis data.	Oktober 14, 2025
Publikasi awal	—	September 30, 2025

AWS Glosarium Panduan Preskriptif

Berikut ini adalah istilah yang umum digunakan dalam strategi, panduan, dan pola yang disediakan oleh Panduan AWS Preskriptif. Untuk menyarankan entri, silakan gunakan tautan Berikan umpan balik di akhir glosarium.

Nomor

7 Rs

Tujuh strategi migrasi umum untuk memindahkan aplikasi ke cloud. Strategi ini dibangun di atas 5 Rs yang diidentifikasi Gartner pada tahun 2011 dan terdiri dari yang berikut:

- Refactor/Re-Architect — Memindahkan aplikasi dan memodifikasi arsitekturnya dengan memanfaatkan sepenuhnya fitur cloud-native untuk meningkatkan kelincahan, kinerja, dan skalabilitas. Ini biasanya melibatkan porting sistem operasi dan database. Contoh: Migrasikan database Oracle lokal Anda ke Amazon Aurora PostgreSQL Compatible Edition.
- Replatform (angkat dan bentuk ulang) — Pindahkan aplikasi ke cloud, dan perkenalkan beberapa tingkat pengoptimalan untuk memanfaatkan kemampuan cloud. Contoh: Migrasikan database Oracle lokal Anda ke Amazon Relational Database Service (Amazon RDS) untuk Oracle di. AWS Cloud
- Pembelian kembali (drop and shop) - Beralih ke produk yang berbeda, biasanya dengan beralih dari lisensi tradisional ke model SaaS. Contoh: Migrasikan sistem manajemen hubungan pelanggan (CRM) Anda ke Salesforce.com.
- Rehost (lift dan shift) — Pindahkan aplikasi ke cloud tanpa membuat perubahan apa pun untuk memanfaatkan kemampuan cloud. Contoh: Migrasikan database Oracle lokal Anda ke Oracle pada instans EC2 di. AWS Cloud
- Relokasi (hypervisor-level lift and shift) — Pindahkan infrastruktur ke cloud tanpa membeli perangkat keras baru, menulis ulang aplikasi, atau memodifikasi operasi yang ada. Anda memigrasikan server dari platform lokal ke layanan cloud untuk platform yang sama. Contoh: Migrasikan Microsoft Hyper-V aplikasi ke AWS.
- Pertahankan (kunjungi kembali) - Simpan aplikasi di lingkungan sumber Anda. Ini mungkin termasuk aplikasi yang memerlukan refactoring besar, dan Anda ingin menunda pekerjaan itu sampai nanti, dan aplikasi lama yang ingin Anda pertahankan, karena tidak ada pembenaran bisnis untuk memigrasikannya.

- Pensiun — Menonaktifkan atau menghapus aplikasi yang tidak lagi diperlukan di lingkungan sumber Anda.

A

ABAC

Lihat kontrol akses berbasis atribut.

layanan abstrak

Lihat layanan terkelola.

ASAM

Lihat atomisitas, konsistensi, isolasi, daya tahan.

migrasi aktif-aktif

Metode migrasi database di mana database sumber dan target tetap sinkron (dengan menggunakan alat replikasi dua arah atau operasi penulisan ganda), dan kedua database menangani transaksi dari menghubungkan aplikasi selama migrasi. Metode ini mendukung migrasi dalam batch kecil yang terkontrol alih-alih memerlukan pemotongan satu kali. Ini lebih fleksibel tetapi membutuhkan lebih banyak pekerjaan daripada migrasi aktif-pasif.

migrasi aktif-pasif

Metode migrasi database di mana database sumber dan target disimpan dalam sinkron, tetapi hanya database sumber yang menangani transaksi dari menghubungkan aplikasi sementara data direplikasi ke database target. Basis data target tidak menerima transaksi apa pun selama migrasi.

fungsi agregat

Fungsi SQL yang beroperasi pada sekelompok baris dan menghitung nilai pengembalian tunggal untuk grup. Contoh fungsi agregat meliputi `SUM` dan`MAX`.

AI

Lihat kecerdasan buatan.

AIOps

Lihat operasi kecerdasan buatan.

anonimisasi

Proses menghapus informasi pribadi secara permanen dalam kumpulan data. Anonimisasi dapat membantu melindungi privasi pribadi. Data anonim tidak lagi dianggap sebagai data pribadi.

anti-pola

Solusi yang sering digunakan untuk masalah berulang di mana solusinya kontra-produktif, tidak efektif, atau kurang efektif daripada alternatif.

kontrol aplikasi

Pendekatan keamanan yang memungkinkan penggunaan hanya aplikasi yang disetujui untuk membantu melindungi sistem dari malware.

portofolio aplikasi

Kumpulan informasi rinci tentang setiap aplikasi yang digunakan oleh organisasi, termasuk biaya untuk membangun dan memelihara aplikasi, dan nilai bisnisnya. Informasi ini adalah kunci untuk penemuan portofolio dan proses analisis dan membantu mengidentifikasi dan memprioritaskan aplikasi yang akan dimigrasi, dimodernisasi, dan dioptimalkan.

kecerdasan buatan (AI)

Bidang ilmu komputer yang didedikasikan untuk menggunakan teknologi komputasi untuk melakukan fungsi kognitif yang biasanya terkait dengan manusia, seperti belajar, memecahkan masalah, dan mengenali pola. Untuk informasi lebih lanjut, lihat Apa itu Kecerdasan Buatan?

operasi kecerdasan buatan (AIOps)

Proses menggunakan teknik pembelajaran mesin untuk memecahkan masalah operasional, mengurangi insiden operasional dan intervensi manusia, dan meningkatkan kualitas layanan. Untuk informasi selengkapnya tentang cara AIOps digunakan dalam strategi AWS migrasi, lihat panduan integrasi operasi.

enkripsi asimetris

Algoritma enkripsi yang menggunakan sepasang kunci, kunci publik untuk enkripsi dan kunci pribadi untuk dekripsi. Anda dapat berbagi kunci publik karena tidak digunakan untuk dekripsi, tetapi akses ke kunci pribadi harus sangat dibatasi.

atomisitas, konsistensi, isolasi, daya tahan (ACID)

Satu set properti perangkat lunak yang menjamin validitas data dan keandalan operasional database, bahkan dalam kasus kesalahan, kegagalan daya, atau masalah lainnya.

kontrol akses berbasis atribut (ABAC)

Praktik membuat izin berbutir halus berdasarkan atribut pengguna, seperti departemen, peran pekerjaan, dan nama tim. Untuk informasi selengkapnya, lihat ABAC untuk AWS dokumentasi AWS Identity and Access Management (IAM).

sumber data otoritatif

Lokasi di mana Anda menyimpan versi utama data, yang dianggap sebagai sumber informasi yang paling dapat diandalkan. Anda dapat menyalin data dari sumber data otoritatif ke lokasi lain untuk tujuan memproses atau memodifikasi data, seperti menganonimkan, menyunting, atau membuat nama samaran.

Zona Ketersediaan

Lokasi berbeda di dalam AWS Region yang terisolasi dari kegagalan di Availability Zone lainnya dan menyediakan konektivitas jaringan latensi rendah yang murah ke Availability Zone lainnya di Wilayah yang sama.

AWS Kerangka Adopsi Cloud (AWS CAF)

Kerangka pedoman dan praktik terbaik AWS untuk membantu organisasi mengembangkan rencana yang efisien dan efektif untuk bergerak dengan sukses ke cloud. AWS CAF mengatur panduan ke dalam enam area fokus yang disebut perspektif: bisnis, orang, tata kelola, platform, keamanan, dan operasi. Perspektif bisnis, orang, dan tata kelola fokus pada keterampilan dan proses bisnis; perspektif platform, keamanan, dan operasi fokus pada keterampilan dan proses teknis. Misalnya, perspektif masyarakat menargetkan pemangku kepentingan yang menangani sumber daya manusia (SDM), fungsi kepegawaian, dan manajemen orang. Untuk perspektif ini, AWS CAF memberikan panduan untuk pengembangan, pelatihan, dan komunikasi orang untuk membantu mempersiapkan organisasi untuk adopsi cloud yang sukses. Untuk informasi lebih lanjut, lihat situs web AWS CAF dan whitepaper AWS CAF.

AWS Kerangka Kualifikasi Beban Kerja (AWS WQF)

Alat yang mengevaluasi beban kerja migrasi database, merekomendasikan strategi migrasi, dan memberikan perkiraan kerja. AWS WQF disertakan dengan AWS Schema Conversion Tool ()AWS SCT. Ini menganalisis skema database dan objek kode, kode aplikasi, dependensi, dan karakteristik kinerja, dan memberikan laporan penilaian.

B

bot buruk

Bot yang dimaksudkan untuk mengganggu atau menyebabkan kerugian bagi individu atau organisasi.

BCP

Lihat perencanaan kontinuitas bisnis.

grafik perilaku

Pandangan interaktif yang terpadu tentang perilaku dan interaksi sumber daya dari waktu ke waktu. Anda dapat menggunakan grafik perilaku dengan Amazon Detective untuk memeriksa upaya logon yang gagal, panggilan API yang mencurigakan, dan tindakan serupa. Untuk informasi selengkapnya, lihat Data dalam grafik perilaku di dokumentasi Detektif.

sistem big-endian

Sistem yang menyimpan byte paling signifikan terlebih dahulu. Lihat juga endianness.

klasifikasi biner

Sebuah proses yang memprediksi hasil biner (salah satu dari dua kelas yang mungkin). Misalnya, model ML Anda mungkin perlu memprediksi masalah seperti “Apakah email ini spam atau bukan spam?” atau “Apakah produk ini buku atau mobil?”

filter mekar

Struktur data probabilistik dan efisien memori yang digunakan untuk menguji apakah suatu elemen adalah anggota dari suatu himpunan.

deployment biru/hijau

Strategi penyebaran tempat Anda membuat dua lingkungan yang terpisah namun identik. Anda menjalankan versi aplikasi saat ini di satu lingkungan (biru) dan versi aplikasi baru di lingkungan lain (hijau). Strategi ini membantu Anda dengan cepat memutar kembali dengan dampak minimal.

bot

Aplikasi perangkat lunak yang menjalankan tugas otomatis melalui internet dan mensimulasikan aktivitas atau interaksi manusia. Beberapa bot berguna atau bermanfaat, seperti perayap web yang mengindeks informasi di internet. Beberapa bot lain, yang dikenal sebagai bot buruk, dimaksudkan untuk mengganggu atau membahayakan individu atau organisasi.

botnet

Jaringan bot yang terinfeksi oleh malware dan berada di bawah kendali satu pihak, yang dikenal sebagai bot herder atau operator bot. Botnet adalah mekanisme paling terkenal untuk skala bot dan dampaknya.

cabang

Area berisi repositori kode. Cabang pertama yang dibuat dalam repositori adalah cabang utama. Anda dapat membuat cabang baru dari cabang yang ada, dan Anda kemudian dapat mengembangkan fitur atau memperbaiki bug di cabang baru. Cabang yang Anda buat untuk membangun fitur biasanya disebut sebagai cabang fitur. Saat fitur siap dirilis, Anda menggabungkan cabang fitur kembali ke cabang utama. Untuk informasi selengkapnya, lihat Tentang cabang (GitHub dokumentasi).

akses break-glass

Dalam keadaan luar biasa dan melalui proses yang disetujui, cara cepat bagi pengguna untuk mendapatkan akses ke Akun AWS yang biasanya tidak memiliki izin untuk mengaksesnya. Untuk informasi lebih lanjut, lihat indikator Implementasikan prosedur break-glass dalam panduan Well-Architected AWS .

strategi brownfield

Infrastruktur yang ada di lingkungan Anda. Saat mengadopsi strategi brownfield untuk arsitektur sistem, Anda merancang arsitektur di sekitar kendala sistem dan infrastruktur saat ini. Jika Anda memperluas infrastruktur yang ada, Anda dapat memadukan strategi brownfield dan greenfield.

cache penyangga

Area memori tempat data yang paling sering diakses disimpan.

kemampuan bisnis

Apa yang dilakukan bisnis untuk menghasilkan nilai (misalnya, penjualan, layanan pelanggan, atau pemasaran). Arsitektur layanan mikro dan keputusan pengembangan dapat didorong oleh kemampuan bisnis. Untuk informasi selengkapnya, lihat bagian Terorganisir di sekitar kemampuan bisnis dari Menjalankan layanan mikro kontainer di whitepaper. AWS

perencanaan kelangsungan bisnis (BCP)

Rencana yang membahas dampak potensial dari peristiwa yang mengganggu, seperti migrasi skala besar, pada operasi dan memungkinkan bisnis untuk melanjutkan operasi dengan cepat.

C

KAFE

Lihat Kerangka Adopsi AWS Cloud.

penyebaran kenari

Rilis versi yang lambat dan bertahap untuk pengguna akhir. Ketika Anda yakin, Anda menyebarkan versi baru dan mengganti versi saat ini secara keseluruhan.

CCoE

Lihat Cloud Center of Excellence.

CDC

Lihat mengubah pengambilan data.

ubah pengambilan data (CDC)

Proses melacak perubahan ke sumber data, seperti tabel database, dan merekam metadata tentang perubahan tersebut. Anda dapat menggunakan CDC untuk berbagai tujuan, seperti mengaudit atau mereplikasi perubahan dalam sistem target untuk mempertahankan sinkronisasi.

rekayasa kekacauan

Sengaja memperkenalkan kegagalan atau peristiwa yang mengganggu untuk menguji ketahanan sistem. Anda dapat menggunakan AWS Fault Injection Service (AWS FIS) untuk melakukan eksperimen yang menekankan AWS beban kerja Anda dan mengevaluasi responsnya.

CI/CD

Lihat integrasi berkelanjutan dan pengiriman berkelanjutan.

klasifikasi

Proses kategorisasi yang membantu menghasilkan prediksi. Model ML untuk masalah klasifikasi memprediksi nilai diskrit. Nilai diskrit selalu berbeda satu sama lain. Misalnya, model mungkin perlu mengevaluasi apakah ada mobil dalam gambar atau tidak.

Enkripsi sisi klien

Enkripsi data secara lokal, sebelum target Layanan AWS menerimanya.

Pusat Keunggulan Cloud (CCoE)

Tim multi-disiplin yang mendorong upaya adopsi cloud di seluruh organisasi, termasuk mengembangkan praktik terbaik cloud, memobilisasi sumber daya, menetapkan jadwal migrasi, dan memimpin organisasi melalui transformasi skala besar. Untuk informasi selengkapnya, lihat posting CCo E di Blog Strategi AWS Cloud Perusahaan.

komputasi cloud

Teknologi cloud yang biasanya digunakan untuk penyimpanan data jarak jauh dan manajemen perangkat IoT. Cloud computing umumnya terhubung ke teknologi edge computing.

model operasi cloud

Dalam organisasi TI, model operasi yang digunakan untuk membangun, mematangkan, dan mengoptimalkan satu atau lebih lingkungan cloud. Untuk informasi selengkapnya, lihat Membangun Model Operasi Cloud Anda.

tahap adopsi cloud

Empat fase yang biasanya dilalui organisasi ketika mereka bermigrasi ke AWS Cloud:

- Proyek — Menjalankan beberapa proyek terkait cloud untuk bukti konsep dan tujuan pembelajaran
- Foundation — Melakukan investasi dasar untuk meningkatkan adopsi cloud Anda (misalnya, membuat landing zone, mendefinisikan CCo E, membuat model operasi)
- Migrasi — Migrasi aplikasi individual
- Re-invention — Mengoptimalkan produk dan layanan, dan berinovasi di cloud

Tahapan ini didefinisikan oleh Stephen Orban dalam posting blog The Journey Toward Cloud-First & the Stages of Adoption di blog Strategi Perusahaan. AWS Cloud Untuk informasi tentang bagaimana kaitannya dengan strategi AWS migrasi, lihat panduan kesiapan migrasi.

CMDB

Lihat database manajemen konfigurasi.

repositori kode

Lokasi di mana kode sumber dan aset lainnya, seperti dokumentasi, sampel, dan skrip, disimpan dan diperbarui melalui proses kontrol versi. Repositori cloud umum termasuk GitHub atau. Bitbucket Cloud Setiap versi kode disebut cabang. Dalam struktur layanan mikro, setiap repositori

dikhususkan untuk satu bagian fungsionalitas. Pipa CI/CD tunggal dapat menggunakan beberapa repositori.

cache dingin

Cache buffer yang kosong, tidak terisi dengan baik, atau berisi data basi atau tidak relevan. Ini mempengaruhi kinerja karena instance database harus membaca dari memori utama atau disk, yang lebih lambat daripada membaca dari cache buffer.

data dingin

Data yang jarang diakses dan biasanya historis. Saat menanyakan jenis data ini, kueri lambat biasanya dapat diterima. Memindahkan data ini ke tingkat penyimpanan atau kelas yang berkinerja lebih rendah dan lebih murah dapat mengurangi biaya.

visi komputer (CV)

Bidang AI yang menggunakan pembelajaran mesin untuk menganalisis dan mengekstrak informasi dari format visual seperti gambar dan video digital. Misalnya, Amazon SageMaker AI menyediakan algoritma pemrosesan gambar untuk CV.

konfigurasi drift

Untuk beban kerja, konfigurasi berubah dari status yang diharapkan. Ini dapat menyebabkan beban kerja menjadi tidak patuh, dan biasanya bertahap dan tidak disengaja.

database manajemen konfigurasi (CMDB)

Repositori yang menyimpan dan mengelola informasi tentang database dan lingkungan TI, termasuk komponen perangkat keras dan perangkat lunak dan konfigurasinya. Anda biasanya menggunakan data dari CMDB dalam penemuan portofolio dan tahap analisis migrasi.

paket kesesuaian

Kumpulan AWS Config aturan dan tindakan remediasi yang dapat Anda kumpulkan untuk menyesuaikan kepatuhan dan pemeriksaan keamanan Anda. Anda dapat menerapkan paket kesesuaian sebagai entitas tunggal di Akun AWS dan Region, atau di seluruh organisasi, dengan menggunakan templat YAMM. Untuk informasi selengkapnya, lihat Paket kesesuaian dalam dokumentasi. AWS Config

integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD)

Proses mengotomatiskan sumber, membangun, menguji, pementasan, dan tahap produksi dari proses rilis perangkat lunak. CI/CD biasanya digambarkan sebagai pipa. CI/CD dapat membantu

Anda mengotomatiskan proses, meningkatkan produktivitas, meningkatkan kualitas kode, dan memberikan lebih cepat. Untuk informasi lebih lanjut, lihat Manfaat pengiriman berkelanjutan. CD juga dapat berarti penerapan berkelanjutan. Untuk informasi selengkapnya, lihat Continuous Delivery vs Continuous Deployment.

CV

Lihat visi komputer.

D

data saat istirahat

Data yang stasioner di jaringan Anda, seperti data yang ada di penyimpanan.

klasifikasi data

Proses untuk mengidentifikasi dan mengkategorikan data dalam jaringan Anda berdasarkan kekritisan dan sensitivitasnya. Ini adalah komponen penting dari setiap strategi manajemen risiko keamanan siber karena membantu Anda menentukan perlindungan dan kontrol retensi yang tepat untuk data. Klasifikasi data adalah komponen pilar keamanan dalam AWS Well-Architected Framework. Untuk informasi selengkapnya, lihat Klasifikasi data.

penyimpangan data

Variasi yang berarti antara data produksi dan data yang digunakan untuk melatih model ML, atau perubahan yang berarti dalam data input dari waktu ke waktu. Penyimpangan data dapat mengurangi kualitas, akurasi, dan keadilan keseluruhan dalam prediksi model ML.

data dalam transit

Data yang aktif bergerak melalui jaringan Anda, seperti antara sumber daya jaringan.

jala data

Kerangka arsitektur yang menyediakan kepemilikan data terdistribusi dan terdesentralisasi dengan manajemen dan tata kelola terpusat.

minimalisasi data

Prinsip pengumpulan dan pemrosesan hanya data yang sangat diperlukan. Mempraktikkan minimalisasi data di dalamnya AWS Cloud dapat mengurangi risiko privasi, biaya, dan jejak karbon analitik Anda.

perimeter data

Satu set pagar pembatas pencegahan di AWS lingkungan Anda yang membantu memastikan bahwa hanya identitas tepercaya yang mengakses sumber daya tepercaya dari jaringan yang diharapkan. Untuk informasi selengkapnya, lihat Membangun perimeter data pada AWS.

prapemrosesan data

Untuk mengubah data mentah menjadi format yang mudah diuraikan oleh model ML Anda. Preprocessing data dapat berarti menghapus kolom atau baris tertentu dan menangani nilai yang hilang, tidak konsisten, atau duplikat.

asal data

Proses melacak asal dan riwayat data sepanjang siklus hidupnya, seperti bagaimana data dihasilkan, ditransmisikan, dan disimpan.

subjek data

Individu yang datanya dikumpulkan dan diproses.

gudang data

Sistem manajemen data yang mendukung intelijen bisnis, seperti analitik. Gudang data biasanya berisi sejumlah besar data historis, dan biasanya digunakan untuk kueri dan analisis.

bahasa definisi database (DDL)

Pernyataan atau perintah untuk membuat atau memodifikasi struktur tabel dan objek dalam database.

bahasa manipulasi basis data (DHTML)

Pernyataan atau perintah untuk memodifikasi (memasukkan, memperbarui, dan menghapus) informasi dalam database.

DDL

Lihat bahasa definisi database.

ansambel yang dalam

Untuk menggabungkan beberapa model pembelajaran mendalam untuk prediksi. Anda dapat menggunakan ansambel dalam untuk mendapatkan prediksi yang lebih akurat atau untuk memperkirakan ketidakpastian dalam prediksi.

pembelajaran mendalam

Subbidang ML yang menggunakan beberapa lapisan jaringan saraf tiruan untuk mengidentifikasi pemetaan antara data input dan variabel target yang diinginkan.

defense-in-depth

Pendekatan keamanan informasi di mana serangkaian mekanisme dan kontrol keamanan dilapisi dengan cermat di seluruh jaringan komputer untuk melindungi kerahasiaan, integritas, dan ketersediaan jaringan dan data di dalamnya. Saat Anda mengadopsi strategi ini AWS, Anda menambahkan beberapa kontrol pada lapisan AWS Organizations struktur yang berbeda untuk membantu mengamankan sumber daya. Misalnya, defense-in-depth pendekatan mungkin menggabungkan otentikasi multi-faktor, segmentasi jaringan, dan enkripsi.

administrator yang didelegasikan

Di AWS Organizations, layanan yang kompatibel dapat mendaftarkan akun AWS anggota untuk mengelola akun organisasi dan mengelola izin untuk layanan tersebut. Akun ini disebut administrator yang didelegasikan untuk layanan itu. Untuk informasi selengkapnya dan daftar layanan yang kompatibel, lihat Layanan yang berfungsi dengan AWS Organizations AWS Organizations dokumentasi.

deployment

Proses pembuatan aplikasi, fitur baru, atau perbaikan kode tersedia di lingkungan target. Deployment melibatkan penerapan perubahan dalam basis kode dan kemudian membangun dan menjalankan basis kode itu di lingkungan aplikasi.

lingkungan pengembangan

Lihat lingkungan.

kontrol detektif

Kontrol keamanan yang dirancang untuk mendeteksi, mencatat, dan memperingatkan setelah suatu peristiwa terjadi. Kontrol ini adalah garis pertahanan kedua, memperingatkan Anda tentang peristiwa keamanan yang melewati kontrol pencegahan yang ada. Untuk informasi selengkapnya, lihat Kontrol Detektif dalam Menerapkan kontrol keamanan pada. AWS

pemetaan aliran nilai pengembangan (DVSM)

Sebuah proses yang digunakan untuk mengidentifikasi dan memprioritaskan kendala yang mempengaruhi kecepatan dan kualitas dalam siklus hidup pengembangan perangkat lunak. DVSM memperluas proses pemetaan aliran nilai yang awalnya dirancang untuk praktik

manufaktur ramping. Ini berfokus pada langkah-langkah dan tim yang diperlukan untuk menciptakan dan memindahkan nilai melalui proses pengembangan perangkat lunak.

kembar digital

Representasi virtual dari sistem dunia nyata, seperti bangunan, pabrik, peralatan industri, atau jalur produksi. Kembar digital mendukung pemeliharaan prediktif, pemantauan jarak jauh, dan optimalisasi produksi.

tabel dimensi

Dalam skema bintang, tabel yang lebih kecil yang berisi atribut data tentang data kuantitatif dalam tabel fakta. Atribut tabel dimensi biasanya bidang teks atau angka diskrit yang berperilaku seperti teks. Atribut ini biasanya digunakan untuk pembatasan kueri, pemfilteran, dan pelabelan set hasil.

musibah

Peristiwa yang mencegah beban kerja atau sistem memenuhi tujuan bisnisnya di lokasi utama yang digunakan. Peristiwa ini dapat berupa bencana alam, kegagalan teknis, atau akibat dari tindakan manusia, seperti kesalahan konfigurasi yang tidak disengaja atau serangan malware.

pemulihan bencana (DR)

Strategi dan proses yang Anda gunakan untuk meminimalkan downtime dan kehilangan data yang disebabkan oleh bencana. Untuk informasi selengkapnya, lihat Disaster Recovery of Workloads on AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML~

Lihat bahasa manipulasi basis data.

desain berbasis domain

Pendekatan untuk mengembangkan sistem perangkat lunak yang kompleks dengan menghubungkan komponennya ke domain yang berkembang, atau tujuan bisnis inti, yang dilayani oleh setiap komponen. Konsep ini diperkenalkan oleh Eric Evans dalam bukunya, Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). Untuk informasi tentang cara menggunakan desain berbasis domain dengan pola gambar pencekik, lihat Memodernisasi layanan web Microsoft ASP.NET (ASMX) lama secara bertahap menggunakan container dan Amazon API Gateway.

DR

Lihat pemulihan bencana.

deteksi drift

Melacak penyimpangan dari konfigurasi dasar. Misalnya, Anda dapat menggunakan AWS CloudFormation untuk mendeteksi penyimpangan dalam sumber daya sistem, atau Anda dapat menggunakannya AWS Control Tower untuk mendeteksi perubahan di landing zone yang mungkin memengaruhi kepatuhan terhadap persyaratan tata kelola.

DVSM

Lihat pemetaan aliran nilai pengembangan.

E

EDA

Lihat analisis data eksplorasi.

EDI

Lihat pertukaran data elektronik.

komputasi tepi

Teknologi yang meningkatkan daya komputasi untuk perangkat pintar di tepi jaringan IoT. Jika dibandingkan dengan komputasi awan, komputasi tepi dapat mengurangi latensi komunikasi dan meningkatkan waktu respons.

pertukaran data elektronik (EDI)

Pertukaran otomatis dokumen bisnis antar organisasi. Untuk informasi selengkapnya, lihat Apa itu Pertukaran Data Elektronik.

enkripsi

Proses komputasi yang mengubah data plaintext, yang dapat dibaca manusia, menjadi ciphertext.

kunci enkripsi

String kriptografi dari bit acak yang dihasilkan oleh algoritma enkripsi. Panjang kunci dapat bervariasi, dan setiap kunci dirancang agar tidak dapat diprediksi dan unik.

endianness

Urutan byte disimpan dalam memori komputer. Sistem big-endian menyimpan byte paling signifikan terlebih dahulu. Sistem little-endian menyimpan byte paling tidak signifikan terlebih dahulu.

titik akhir

Lihat titik akhir layanan.

layanan endpoint

Layanan yang dapat Anda host di cloud pribadi virtual (VPC) untuk dibagikan dengan pengguna lain. Anda dapat membuat layanan endpoint dengan AWS PrivateLink dan memberikan izin kepada prinsipal lain Akun AWS atau ke AWS Identity and Access Management (IAM). Akun atau prinsipal ini dapat terhubung ke layanan endpoint Anda secara pribadi dengan membuat titik akhir VPC antarmuka. Untuk informasi selengkapnya, lihat Membuat layanan titik akhir di dokumentasi Amazon Virtual Private Cloud (Amazon VPC).

perencanaan sumber daya perusahaan (ERP)

Sistem yang mengotomatiskan dan mengelola proses bisnis utama (seperti akuntansi, MES, dan manajemen proyek) untuk suatu perusahaan.

enkripsi amplop

Proses mengenkripsi kunci enkripsi dengan kunci enkripsi lain. Untuk informasi selengkapnya, lihat Enkripsi amplop dalam dokumentasi AWS Key Management Service (AWS KMS).

lingkungan

Sebuah contoh dari aplikasi yang sedang berjalan. Berikut ini adalah jenis lingkungan yang umum dalam komputasi awan:

- Development Environment — Sebuah contoh dari aplikasi yang berjalan yang hanya tersedia untuk tim inti yang bertanggung jawab untuk memelihara aplikasi. Lingkungan pengembangan digunakan untuk menguji perubahan sebelum mempromosikannya ke lingkungan atas. Jenis lingkungan ini kadang-kadang disebut sebagai lingkungan pengujian.
- lingkungan yang lebih rendah — Semua lingkungan pengembangan untuk aplikasi, seperti yang digunakan untuk build awal dan pengujian.
- lingkungan produksi — Sebuah contoh dari aplikasi yang berjalan yang dapat diakses oleh pengguna akhir. Dalam sebuah CI/CD pipeline, lingkungan produksi adalah lingkungan penyebaran terakhir.
- lingkungan atas — Semua lingkungan yang dapat diakses oleh pengguna selain tim pengembangan inti. Ini dapat mencakup lingkungan produksi, lingkungan praproduksi, dan lingkungan untuk pengujian penerimaan pengguna.

epik

Dalam metodologi tangkas, kategori fungsional yang membantu mengatur dan memprioritaskan pekerjaan Anda. Epik memberikan deskripsi tingkat tinggi tentang persyaratan dan tugas implementasi. Misalnya, epos keamanan AWS CAF mencakup manajemen identitas dan akses, kontrol detektif, keamanan infrastruktur, perlindungan data, dan respons insiden. Untuk informasi selengkapnya tentang epos dalam strategi AWS migrasi, lihat panduan implementasi program.

ERP

Lihat perencanaan sumber daya perusahaan.

analisis data eksplorasi (EDA)

Proses menganalisis dataset untuk memahami karakteristik utamanya. Anda mengumpulkan atau mengumpulkan data dan kemudian melakukan penyelidikan awal untuk menemukan pola, mendeteksi anomali, dan memeriksa asumsi. EDA dilakukan dengan menghitung statistik ringkasan dan membuat visualisasi data.

F

tabel fakta

Tabel tengah dalam skema bintang. Ini menyimpan data kuantitatif tentang operasi bisnis. Biasanya, tabel fakta berisi dua jenis kolom: kolom yang berisi ukuran dan yang berisi kunci asing ke tabel dimensi.

gagal cepat

Filosofi yang menggunakan pengujian yang sering dan bertahap untuk mengurangi siklus hidup pengembangan. Ini adalah bagian penting dari pendekatan tangkas.

batas isolasi kesalahan

Dalam AWS Cloud, batas seperti Availability Zone, AWS Region, control plane, atau data plane yang membatasi efek kegagalan dan membantu meningkatkan ketahanan beban kerja. Untuk informasi selengkapnya, lihat Batas Isolasi AWS Kesalahan.

cabang fitur

Lihat cabang.

fitur

Data input yang Anda gunakan untuk membuat prediksi. Misalnya, dalam konteks manufaktur, fitur bisa berupa gambar yang diambil secara berkala dari lini manufaktur.

pentingnya fitur

Seberapa signifikan fitur untuk prediksi model. Ini biasanya dinyatakan sebagai skor numerik yang dapat dihitung melalui berbagai teknik, seperti Shapley Additive Explanations (SHAP) dan gradien terintegrasi. Untuk informasi lebih lanjut, lihat Interpretabilitas model pembelajaran mesin dengan. AWS

transformasi fitur

Untuk mengoptimalkan data untuk proses ML, termasuk memperkaya data dengan sumber tambahan, menskalakan nilai, atau mengekstrak beberapa set informasi dari satu bidang data. Hal ini memungkinkan model ML untuk mendapatkan keuntungan dari data. Misalnya, jika Anda memecah tanggal “2021-05-27 00:15:37" menjadi “2021", “Mei”, “Kamis”, dan “15", Anda dapat membantu algoritme pembelajaran mempelajari pola bernuansa yang terkait dengan komponen data yang berbeda.

beberapa tembakan mendorong

Menyediakan LLM dengan sejumlah kecil contoh yang menunjukkan tugas dan output yang diinginkan sebelum memintanya untuk melakukan tugas serupa. Teknik ini adalah aplikasi pembelajaran dalam konteks, di mana model belajar dari contoh (bidikan) yang tertanam dalam petunjuk. Beberapa bidikan dapat efektif untuk tugas-tugas yang memerlukan pemformatan, penalaran, atau pengetahuan domain tertentu. Lihat juga bidikan nol.

FGAC

Lihat kontrol akses berbutir halus.

kontrol akses berbutir halus (FGAC)

Penggunaan beberapa kondisi untuk mengizinkan atau menolak permintaan akses.

migrasi flash-cut

Metode migrasi database yang menggunakan replikasi data berkelanjutan melalui pengambilan data perubahan untuk memigrasikan data dalam waktu sesingkat mungkin, alih-alih menggunakan pendekatan bertahap. Tujuannya adalah untuk menjaga downtime seminimal mungkin.

FM

Lihat model pondasi.

model pondasi (FM)

Jaringan saraf pembelajaran mendalam yang besar yang telah melatih kumpulan data besar-besaran data umum dan tidak berlabel. FMs mampu melakukan berbagai tugas umum, seperti memahami bahasa, menghasilkan teks dan gambar, dan berbicara dalam bahasa alami. Untuk informasi selengkapnya, lihat Apa itu Model Foundation.

G

AI generatif

Subset model AI yang telah dilatih pada sejumlah besar data dan yang dapat menggunakan prompt teks sederhana untuk membuat konten dan artefak baru, seperti gambar, video, teks, dan audio. Untuk informasi lebih lanjut, lihat Apa itu AI Generatif.

pemblokiran geografis

Lihat pembatasan geografis.

pembatasan geografis (pemblokiran geografis)

Di Amazon CloudFront, opsi untuk mencegah pengguna di negara tertentu mengakses distribusi konten. Anda dapat menggunakan daftar izinkan atau daftar blokir untuk menentukan negara yang disetujui dan dilarang. Untuk informasi selengkapnya, lihat Membatasi distribusi geografis konten Anda dalam dokumentasi. CloudFront

Alur kerja Gitflow

Pendekatan di mana lingkungan bawah dan atas menggunakan cabang yang berbeda dalam repositori kode sumber. Alur kerja Gitflow dianggap warisan, dan alur kerja berbasis batang adalah pendekatan modern yang lebih disukai.

gambar emas

Sebuah snapshot dari sistem atau perangkat lunak yang digunakan sebagai template untuk menyebarkan instance baru dari sistem atau perangkat lunak itu. Misalnya, di bidang manufaktur, gambar emas dapat digunakan untuk menyediakan perangkat lunak pada beberapa perangkat dan membantu meningkatkan kecepatan, skalabilitas, dan produktivitas dalam operasi manufaktur perangkat.

strategi greenfield

Tidak adanya infrastruktur yang ada di lingkungan baru. Saat mengadopsi strategi greenfield untuk arsitektur sistem, Anda dapat memilih semua teknologi baru tanpa batasan kompatibilitas dengan infrastruktur yang ada, juga dikenal sebagai brownfield. Jika Anda memperluas infrastruktur yang ada, Anda dapat memadukan strategi brownfield dan greenfield.

pagar pembatas

Aturan tingkat tinggi yang membantu mengatur sumber daya, kebijakan, dan kepatuhan di seluruh unit organisasi ()OUs. Pagar pembatas preventif menegakkan kebijakan untuk memastikan keselarasan dengan standar kepatuhan. Mereka diimplementasikan dengan menggunakan kebijakan kontrol layanan dan batas izin IAM. Detective guardrails mendeteksi pelanggaran kebijakan dan masalah kepatuhan, dan menghasilkan peringatan untuk remediasi. Mereka diimplementasikan dengan menggunakan AWS Config, AWS Security Hub CSPM, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector, dan pemeriksaan khusus AWS Lambda .

H

HA

Lihat ketersediaan tinggi.

migrasi database heterogen

Memigrasi database sumber Anda ke database target yang menggunakan mesin database yang berbeda (misalnya, Oracle ke Amazon Aurora). Migrasi heterogen biasanya merupakan bagian dari upaya arsitektur ulang, dan mengubah skema dapat menjadi tugas yang kompleks. AWS menyediakan AWS SCT yang membantu dengan konversi skema.

ketersediaan tinggi (HA)

Kemampuan beban kerja untuk beroperasi terus menerus, tanpa intervensi, jika terjadi tantangan atau bencana. Sistem HA dirancang untuk gagal secara otomatis, secara konsisten memberikan kinerja berkualitas tinggi, dan menangani beban dan kegagalan yang berbeda dengan dampak kinerja minimal.

modernisasi sejarawan

Pendekatan yang digunakan untuk memodernisasi dan meningkatkan sistem teknologi operasional (OT) untuk melayani kebutuhan industri manufaktur dengan lebih baik. Sejarawan

adalah jenis database yang digunakan untuk mengumpulkan dan menyimpan data dari berbagai sumber di pabrik.

data penahanan

Sebagian dari data historis berlabel yang ditahan dari kumpulan data yang digunakan untuk melatih model pembelajaran mesin. Anda dapat menggunakan data penahanan untuk mengevaluasi kinerja model dengan membandingkan prediksi model dengan data penahanan.

migrasi database homogen

Memigrasi database sumber Anda ke database target yang berbagi mesin database yang sama (misalnya, Microsoft SQL Server ke Amazon RDS for SQL Server). Migrasi homogen biasanya merupakan bagian dari upaya rehosting atau replatforming. Anda dapat menggunakan utilitas database asli untuk memigrasi skema.

data panas

Data yang sering diakses, seperti data real-time atau data translasi terbaru. Data ini biasanya memerlukan tingkat atau kelas penyimpanan berkinerja tinggi untuk memberikan respons kueri yang cepat.

perbaikan terbaru

Perbaikan mendesak untuk masalah kritis dalam lingkungan produksi. Karena urgensinya, perbaikan terbaru biasanya dibuat di luar alur kerja DevOps rilis biasa.

periode hypercare

Segera setelah cutover, periode waktu ketika tim migrasi mengelola dan memantau aplikasi yang dimigrasi di cloud untuk mengatasi masalah apa pun. Biasanya, periode ini panjangnya 1-4 hari. Pada akhir periode hypercare, tim migrasi biasanya mentransfer tanggung jawab untuk aplikasi ke tim operasi cloud.

I

IAc

Lihat infrastruktur sebagai kode.

kebijakan berbasis identitas

Kebijakan yang dilampirkan pada satu atau beberapa prinsip IAM yang mendefinisikan izin mereka dalam lingkungan. AWS Cloud

aplikasi idle

Aplikasi yang memiliki penggunaan CPU dan memori rata-rata antara 5 dan 20 persen selama periode 90 hari. Dalam proyek migrasi, adalah umum untuk menghentikan aplikasi ini atau mempertahankannya di tempat.

IIoT

Lihat Internet of Things industri.

infrastruktur yang tidak dapat diubah

Model yang menyebarkan infrastruktur baru untuk beban kerja produksi alih-alih memperbarui, menambal, atau memodifikasi infrastruktur yang ada. Infrastruktur yang tidak dapat diubah secara inheren lebih konsisten, andal, dan dapat diprediksi daripada infrastruktur yang dapat berubah. Untuk informasi selengkapnya, lihat praktik terbaik Deploy using immutable infrastructure di AWS Well-Architected Framework.

masuk (masuknya) VPC

Dalam arsitektur AWS multi-akun, VPC yang menerima, memeriksa, dan merutekan koneksi jaringan dari luar aplikasi. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan inbound, outbound, dan inspeksi VPCs untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

migrasi inkremental

Strategi cutover di mana Anda memigrasikan aplikasi Anda dalam bagian-bagian kecil alih-alih melakukan satu cutover penuh. Misalnya, Anda mungkin hanya memindahkan beberapa layanan mikro atau pengguna ke sistem baru pada awalnya. Setelah Anda memverifikasi bahwa semuanya berfungsi dengan baik, Anda dapat secara bertahap memindahkan layanan mikro atau pengguna tambahan hingga Anda dapat menonaktifkan sistem lama Anda. Strategi ini mengurangi risiko yang terkait dengan migrasi besar.

Industri 4.0

Sebuah istilah yang diperkenalkan oleh Klaus Schwab pada tahun 2016 untuk merujuk pada modernisasi proses manufaktur melalui kemajuan dalam konektivitas, data real-time, otomatisasi, analitik, dan AI/ML.

infrastruktur

Semua sumber daya dan aset yang terkandung dalam lingkungan aplikasi.

infrastruktur sebagai kode (IAc)

Proses penyediaan dan pengelolaan infrastruktur aplikasi melalui satu set file konfigurasi. IAc dirancang untuk membantu Anda memusatkan manajemen infrastruktur, menstandarisasi sumber daya, dan menskalakan dengan cepat sehingga lingkungan baru dapat diulang, andal, dan konsisten.

Internet of Things industri (IIoT)

Penggunaan sensor dan perangkat yang terhubung ke internet di sektor industri, seperti manufaktur, energi, otomotif, perawatan kesehatan, ilmu kehidupan, dan pertanian. Untuk informasi lebih lanjut, lihat Membangun strategi transformasi digital Internet of Things (IIoT) industri.

inspeksi VPC

Dalam arsitektur AWS multi-akun, VPC terpusat yang mengelola inspeksi lalu lintas jaringan antara VPCs (dalam yang sama atau berbeda Wilayah AWS), internet, dan jaringan lokal. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan inbound, outbound, dan inspeksi VPCs untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

Internet of Things (IoT)

Jaringan objek fisik yang terhubung dengan sensor atau prosesor tertanam yang berkomunikasi dengan perangkat dan sistem lain melalui internet atau melalui jaringan komunikasi lokal. Untuk informasi selengkapnya, lihat Apa itu IoT?

interpretabilitas

Karakteristik model pembelajaran mesin yang menggambarkan sejauh mana manusia dapat memahami bagaimana prediksi model bergantung pada inputnya. Untuk informasi lebih lanjut, lihat Interpretabilitas model pembelajaran mesin dengan. AWS

IoT

Lihat Internet of Things.

Perpustakaan informasi TI (ITIL)

Serangkaian praktik terbaik untuk memberikan layanan TI dan menyelaraskan layanan ini dengan persyaratan bisnis. ITIL menyediakan dasar untuk ITSM.

Manajemen layanan TI (ITSM)

Kegiatan yang terkait dengan merancang, menerapkan, mengelola, dan mendukung layanan TI untuk suatu organisasi. Untuk informasi tentang mengintegrasikan operasi cloud dengan alat ITSM, lihat panduan integrasi operasi.

ITIL

Lihat perpustakaan informasi TI.

ITSM

Lihat manajemen layanan TI.

L

kontrol akses berbasis label (LBAC)

Implementasi kontrol akses wajib (MAC) di mana pengguna dan data itu sendiri masing-masing secara eksplisit diberi nilai label keamanan. Persimpangan antara label keamanan pengguna dan label keamanan data menentukan baris dan kolom mana yang dapat dilihat oleh pengguna.

landing zone

Landing zone adalah AWS lingkungan multi-akun yang dirancang dengan baik yang dapat diskalakan dan aman. Ini adalah titik awal dari mana organisasi Anda dapat dengan cepat meluncurkan dan menyebarkan beban kerja dan aplikasi dengan percaya diri dalam lingkungan keamanan dan infrastruktur mereka. Untuk informasi selengkapnya tentang zona pendaratan, lihat Menyiapkan lingkungan multi-akun AWS yang aman dan dapat diskalakan.

model bahasa besar (LLM)

Model AI pembelajaran mendalam yang dilatih sebelumnya pada sejumlah besar data. LLM dapat melakukan beberapa tugas, seperti menjawab pertanyaan, meringkas dokumen, menerjemahkan teks ke dalam bahasa lain, dan menyelesaikan kalimat. Untuk informasi lebih lanjut, lihat Apa itu LLMs.

migrasi besar

Migrasi 300 atau lebih server.

LBAC

Lihat kontrol akses berbasis label.

hak istimewa paling sedikit

Praktik keamanan terbaik untuk memberikan izin minimum yang diperlukan untuk melakukan tugas. Untuk informasi selengkapnya, lihat Menerapkan izin hak istimewa terkecil dalam dokumentasi IAM.

angkat dan geser

Lihat 7 Rs.

sistem endian kecil

Sebuah sistem yang menyimpan byte paling tidak signifikan terlebih dahulu. Lihat juga endianness.

LLM

Lihat model bahasa besar.

lingkungan yang lebih rendah

Lihat lingkungan.

M

pembelajaran mesin (ML)

Jenis kecerdasan buatan yang menggunakan algoritma dan teknik untuk pengenalan pola dan pembelajaran. ML menganalisis dan belajar dari data yang direkam, seperti data Internet of Things (IoT), untuk menghasilkan model statistik berdasarkan pola. Untuk informasi selengkapnya, lihat Machine Learning.

cabang utama

Lihat cabang.

malware

Perangkat lunak yang dirancang untuk membahayakan keamanan atau privasi komputer. Malware dapat mengganggu sistem komputer, membocorkan informasi sensitif, atau mendapatkan akses yang tidak sah. Contoh malware termasuk virus, worm, ransomware, Trojan horse, spyware, dan keyloggers.

layanan terkelola

Layanan AWS yang AWS mengoperasikan lapisan infrastruktur, sistem operasi, dan platform, dan Anda mengakses titik akhir untuk menyimpan dan mengambil data. Amazon Simple Storage Service (Amazon S3) dan Amazon DynamoDB adalah contoh layanan terkelola. Ini juga dikenal sebagai layanan abstrak.

sistem eksekusi manufaktur (MES)

Sistem perangkat lunak untuk melacak, memantau, mendokumentasikan, dan mengendalikan proses produksi yang mengubah bahan baku menjadi produk jadi di lantai toko.

PETA

Lihat Program Percepatan Migrasi.

mekanisme

Proses lengkap di mana Anda membuat alat, mendorong adopsi alat, dan kemudian memeriksa hasilnya untuk melakukan penyesuaian. Mekanisme adalah siklus yang memperkuat dan meningkatkan dirinya sendiri saat beroperasi. Untuk informasi lebih lanjut, lihat Membangun mekanisme di AWS Well-Architected Framework.

akun anggota

Semua Akun AWS selain akun manajemen yang merupakan bagian dari organisasi di AWS Organizations. Akun dapat menjadi anggota dari hanya satu organisasi pada suatu waktu.

MES

Lihat sistem eksekusi manufaktur.

Transportasi Telemetri Antrian Pesan (MQTT)

Protokol komunikasi ringan machine-to-machine (M2M), berdasarkan pola terbitkan/berlangganan, untuk perangkat IoT yang dibatasi sumber daya.

layanan mikro

Layanan kecil dan independen yang berkomunikasi dengan jelas APIs dan biasanya dimiliki oleh tim kecil yang mandiri. Misalnya, sistem asuransi mungkin mencakup layanan mikro yang memetakan kemampuan bisnis, seperti penjualan atau pemasaran, atau subdomain, seperti pembelian, klaim, atau analitik. Manfaat layanan mikro termasuk kelincahan, penskalaan yang fleksibel, penyebaran yang mudah, kode yang dapat digunakan kembali, dan ketahanan. Untuk

informasi selengkapnya, lihat Mengintegrasikan layanan mikro dengan menggunakan layanan tanpa AWS server.

arsitektur microservices

Pendekatan untuk membangun aplikasi dengan komponen independen yang menjalankan setiap proses aplikasi sebagai layanan mikro. Layanan mikro ini berkomunikasi melalui antarmuka yang terdefinisi dengan baik dengan menggunakan ringan. APIs Setiap layanan mikro dalam arsitektur ini dapat diperbarui, digunakan, dan diskalakan untuk memenuhi permintaan fungsi tertentu dari suatu aplikasi. Untuk informasi selengkapnya, lihat Menerapkan layanan mikro di AWS.

Program Percepatan Migrasi (MAP)

AWS Program yang menyediakan dukungan konsultasi, pelatihan, dan layanan untuk membantu organisasi membangun fondasi operasional yang kuat untuk pindah ke cloud, dan untuk membantu mengimbangi biaya awal migrasi. MAP mencakup metodologi migrasi untuk mengeksekusi migrasi lama dengan cara metodis dan seperangkat alat untuk mengotomatisasi dan mempercepat skenario migrasi umum.

migrasi dalam skala

Proses memindahkan sebagian besar portofolio aplikasi ke cloud dalam gelombang, dengan lebih banyak aplikasi bergerak pada tingkat yang lebih cepat di setiap gelombang. Fase ini menggunakan praktik dan pelajaran terbaik dari fase sebelumnya untuk mengimplementasikan pabrik migrasi tim, alat, dan proses untuk merampingkan migrasi beban kerja melalui otomatisasi dan pengiriman tangkas. Ini adalah fase ketiga dari strategi AWS migrasi.

pabrik migrasi

Tim lintas fungsi yang merampingkan migrasi beban kerja melalui pendekatan otomatis dan gesit. Tim pabrik migrasi biasanya mencakup operasi, analis dan pemilik bisnis, insinyur migrasi, pengembang, dan DevOps profesional yang bekerja di sprint. Antara 20 dan 50 persen portofolio aplikasi perusahaan terdiri dari pola berulang yang dapat dioptimalkan dengan pendekatan pabrik. Untuk informasi selengkapnya, lihat diskusi tentang pabrik migrasi dan panduan Pabrik Migrasi Cloud di kumpulan konten ini.

metadata migrasi

Informasi tentang aplikasi dan server yang diperlukan untuk menyelesaikan migrasi. Setiap pola migrasi memerlukan satu set metadata migrasi yang berbeda. Contoh metadata migrasi termasuk subnet target, grup keamanan, dan akun. AWS

pola migrasi

Tugas migrasi berulang yang merinci strategi migrasi, tujuan migrasi, dan aplikasi atau layanan migrasi yang digunakan. Contoh: Rehost migrasi ke Amazon EC2 AWS dengan Layanan Migrasi Aplikasi.

Penilaian Portofolio Migrasi (MPA)

Alat online yang menyediakan informasi untuk memvalidasi kasus bisnis untuk bermigrasi ke. AWS Cloud MPA menyediakan penilaian portofolio terperinci (ukuran kanan server, harga, perbandingan TCO, analisis biaya migrasi) serta perencanaan migrasi (analisis data aplikasi dan pengumpulan data, pengelompokan aplikasi, prioritas migrasi, dan perencanaan gelombang). Alat MPA (memerlukan login) tersedia gratis untuk semua AWS konsultan dan konsultan APN Partner.

Penilaian Kesiapan Migrasi (MRA)

Proses mendapatkan wawasan tentang status kesiapan cloud organisasi, mengidentifikasi kekuatan dan kelemahan, dan membangun rencana aksi untuk menutup kesenjangan yang diidentifikasi, menggunakan CAF. AWS Untuk informasi selengkapnya, lihat panduan kesiapan migrasi. MRA adalah tahap pertama dari strategi AWS migrasi.

strategi migrasi

Pendekatan yang digunakan untuk memigrasikan beban kerja ke. AWS Cloud Untuk informasi lebih lanjut, lihat entri 7 Rs di glosarium ini dan lihat Memobilisasi organisasi Anda untuk mempercepat migrasi skala besar.

ML

Lihat pembelajaran mesin.

modernisasi

Mengubah aplikasi usang (warisan atau monolitik) dan infrastrukturnya menjadi sistem yang gesit, elastis, dan sangat tersedia di cloud untuk mengurangi biaya, mendapatkan efisiensi, dan memanfaatkan inovasi. Untuk informasi selengkapnya, lihat Strategi untuk memodernisasi aplikasi di. AWS Cloud

penilaian kesiapan modernisasi

Evaluasi yang membantu menentukan kesiapan modernisasi aplikasi organisasi; mengidentifikasi manfaat, risiko, dan dependensi; dan menentukan seberapa baik organisasi dapat mendukung keadaan masa depan aplikasi tersebut. Hasil penilaian adalah cetak biru arsitektur target, peta

jalan yang merinci fase pengembangan dan tonggak untuk proses modernisasi, dan rencana aksi untuk mengatasi kesenjangan yang diidentifikasi. Untuk informasi lebih lanjut, lihat Mengevaluasi kesiapan modernisasi untuk aplikasi di. AWS Cloud

aplikasi monolitik (monolit)

Aplikasi yang berjalan sebagai layanan tunggal dengan proses yang digabungkan secara ketat. Aplikasi monolitik memiliki beberapa kelemahan. Jika satu fitur aplikasi mengalami lonjakan permintaan, seluruh arsitektur harus diskalakan. Menambahkan atau meningkatkan fitur aplikasi monolitik juga menjadi lebih kompleks ketika basis kode tumbuh. Untuk mengatasi masalah ini, Anda dapat menggunakan arsitektur microservices. Untuk informasi lebih lanjut, lihat Mengurai monolit menjadi layanan mikro.

MPA

Lihat Penilaian Portofolio Migrasi.

MQTT

Lihat Transportasi Telemetri Antrian Pesan.

klasifikasi multiclass

Sebuah proses yang membantu menghasilkan prediksi untuk beberapa kelas (memprediksi satu dari lebih dari dua hasil). Misalnya, model ML mungkin bertanya "Apakah produk ini buku, mobil, atau telepon?" atau "Kategori produk mana yang paling menarik bagi pelanggan ini?"

infrastruktur yang bisa berubah

Model yang memperbarui dan memodifikasi infrastruktur yang ada untuk beban kerja produksi. Untuk meningkatkan konsistensi, keandalan, dan prediktabilitas, AWS Well-Architected Framework merekomendasikan penggunaan infrastruktur yang tidak dapat diubah sebagai praktik terbaik.

O

OAC

Lihat kontrol akses asal.

OAI

Lihat identitas akses asal.

OCM

Lihat manajemen perubahan organisasi.

migrasi offline

Metode migrasi di mana beban kerja sumber diturunkan selama proses migrasi. Metode ini melibatkan waktu henti yang diperpanjang dan biasanya digunakan untuk beban kerja kecil dan tidak kritis.

OI

Lihat integrasi operasi.

OLA

Lihat perjanjian tingkat operasional.

migrasi online

Metode migrasi di mana beban kerja sumber disalin ke sistem target tanpa diambil offline. Aplikasi yang terhubung ke beban kerja dapat terus berfungsi selama migrasi. Metode ini melibatkan waktu henti nol hingga minimal dan biasanya digunakan untuk beban kerja produksi yang kritis.

OPC-UA

Lihat Komunikasi Proses Terbuka - Arsitektur Terpadu.

Komunikasi Proses Terbuka - Arsitektur Terpadu (OPC-UA)

Protokol komunikasi machine-to-machine (M2M) untuk otomasi industri. OPC-UA menyediakan standar interoperabilitas dengan enkripsi data, otentikasi, dan skema otorisasi.

perjanjian tingkat operasional (OLA)

Perjanjian yang menjelaskan apa yang dijanjikan kelompok TI fungsional untuk diberikan satu sama lain, untuk mendukung perjanjian tingkat layanan (SLA).

Tinjauan Kesiapan Operasional (ORR)

Daftar pertanyaan dan praktik terbaik terkait yang membantu Anda memahami, mengevaluasi, mencegah, atau mengurangi ruang lingkup insiden dan kemungkinan kegagalan. Untuk informasi lebih lanjut, lihat Ulasan Kesiapan Operasional (ORR) dalam Kerangka Kerja Well-Architected AWS .

teknologi operasional (OT)

Sistem perangkat keras dan perangkat lunak yang bekerja dengan lingkungan fisik untuk mengendalikan operasi industri, peralatan, dan infrastruktur. Di bidang manufaktur, integrasi sistem OT dan teknologi informasi (TI) adalah fokus utama untuk transformasi Industri 4.0.

integrasi operasi (OI)

Proses modernisasi operasi di cloud, yang melibatkan perencanaan kesiapan, otomatisasi, dan integrasi. Untuk informasi selengkapnya, lihat panduan integrasi operasi.

jejak organisasi

Jejak yang dibuat oleh AWS CloudTrail itu mencatat semua peristiwa untuk semua Akun AWS dalam organisasi di AWS Organizations. Jejak ini dibuat di setiap Akun AWS bagian organisasi dan melacak aktivitas di setiap akun. Untuk informasi selengkapnya, lihat Membuat jejak untuk organisasi dalam CloudTrail dokumentasi.

manajemen perubahan organisasi (OCM)

Kerangka kerja untuk mengelola transformasi bisnis utama yang mengganggu dari perspektif orang, budaya, dan kepemimpinan. OCM membantu organisasi mempersiapkan, dan transisi ke, sistem dan strategi baru dengan mempercepat adopsi perubahan, mengatasi masalah transisi, dan mendorong perubahan budaya dan organisasi. Dalam strategi AWS migrasi, kerangka kerja ini disebut percepatan orang, karena kecepatan perubahan yang diperlukan dalam proyek adopsi cloud. Untuk informasi lebih lanjut, lihat panduan OCM.

kontrol akses asal (OAC)

Di CloudFront, opsi yang disempurnakan untuk membatasi akses untuk mengamankan konten Amazon Simple Storage Service (Amazon S3) Anda. OAC mendukung semua bucket S3 di semua Wilayah AWS, enkripsi sisi server dengan AWS KMS (SSE-KMS), dan dinamis dan permintaan ke bucket S3. `PUT` `DELETE`

identitas akses asal (OAI)

Di CloudFront, opsi untuk membatasi akses untuk mengamankan konten Amazon S3 Anda. Saat Anda menggunakan OAI, CloudFront buat prinsipal yang dapat diautentikasi oleh Amazon S3. Prinsipal yang diautentikasi dapat mengakses konten dalam bucket S3 hanya melalui distribusi tertentu. CloudFront Lihat juga OAC, yang menyediakan kontrol akses yang lebih terperinci dan ditingkatkan.

ORR

Lihat tinjauan kesiapan operasional.

OT

Lihat teknologi operasional.

keluar (jalan keluar) VPC

Dalam arsitektur AWS multi-akun, VPC yang menangani koneksi jaringan yang dimulai dari dalam aplikasi. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan inbound, outbound, dan inspeksi VPCs untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

P

batas izin

Kebijakan manajemen IAM yang dilampirkan pada prinsipal IAM untuk menetapkan izin maksimum yang dapat dimiliki pengguna atau peran. Untuk informasi selengkapnya, lihat Batas izin dalam dokumentasi IAM.

Informasi Identifikasi Pribadi (PII)

Informasi yang, jika dilihat secara langsung atau dipasangkan dengan data terkait lainnya, dapat digunakan untuk menyimpulkan identitas individu secara wajar. Contoh PII termasuk nama, alamat, dan informasi kontak.

PII

Lihat informasi yang dapat diidentifikasi secara pribadi.

buku pedoman

Serangkaian langkah yang telah ditentukan sebelumnya yang menangkap pekerjaan yang terkait dengan migrasi, seperti mengirimkan fungsi operasi inti di cloud. Buku pedoman dapat berupa skrip, runbook otomatis, atau ringkasan proses atau langkah-langkah yang diperlukan untuk mengoperasikan lingkungan modern Anda.

PLC

Lihat pengontrol logika yang dapat diprogram.

PLM

Lihat manajemen siklus hidup produk.

kebijakan

Objek yang dapat menentukan izin (lihat kebijakan berbasis identitas), menentukan kondisi akses (lihat kebijakan berbasis sumber daya), atau menentukan izin maksimum untuk semua akun di organisasi (lihat kebijakan kontrol layanan). AWS Organizations

ketekunan poliglot

Secara independen memilih teknologi penyimpanan data microservice berdasarkan pola akses data dan persyaratan lainnya. Jika layanan mikro Anda memiliki teknologi penyimpanan data yang sama, mereka dapat menghadapi tantangan implementasi atau mengalami kinerja yang buruk. Layanan mikro lebih mudah diimplementasikan dan mencapai kinerja dan skalabilitas yang lebih baik jika mereka menggunakan penyimpanan data yang paling sesuai dengan kebutuhan mereka.

penilaian portofolio

Proses menemukan, menganalisis, dan memprioritaskan portofolio aplikasi untuk merencanakan migrasi. Untuk informasi selengkapnya, lihat Mengevaluasi kesiapan migrasi.

predikat

Kondisi kueri yang mengembalikan `true` atau`false`, biasanya terletak di `WHERE` klausa.

predikat pushdown

Teknik pengoptimalan kueri database yang menyaring data dalam kueri sebelum transfer. Ini mengurangi jumlah data yang harus diambil dan diproses dari database relasional, dan meningkatkan kinerja kueri.

kontrol preventif

Kontrol keamanan yang dirancang untuk mencegah suatu peristiwa terjadi. Kontrol ini adalah garis pertahanan pertama untuk membantu mencegah akses tidak sah atau perubahan yang tidak diinginkan ke jaringan Anda. Untuk informasi selengkapnya, lihat Kontrol pencegahan dalam Menerapkan kontrol keamanan pada. AWS

principal

Entitas AWS yang dapat melakukan tindakan dan mengakses sumber daya. Entitas ini biasanya merupakan pengguna root untuk Akun AWS, peran IAM, atau pengguna. Untuk informasi selengkapnya, lihat Prinsip dalam istilah dan konsep Peran dalam dokumentasi IAM.

privasi berdasarkan desain

Pendekatan rekayasa sistem yang memperhitungkan privasi melalui seluruh proses pengembangan.

zona yang dihosting pribadi

Container yang menyimpan informasi tentang bagaimana Anda ingin Amazon Route 53 merespons kueri DNS untuk domain dan subdomainnya dalam satu atau lebih. VPCs Untuk informasi selengkapnya, lihat Bekerja dengan zona yang dihosting pribadi di dokumentasi Route 53.

kontrol proaktif

Kontrol keamanan yang dirancang untuk mencegah penyebaran sumber daya yang tidak sesuai. Kontrol ini memindai sumber daya sebelum disediakan. Jika sumber daya tidak sesuai dengan kontrol, maka itu tidak disediakan. Untuk informasi selengkapnya, lihat panduan referensi Kontrol dalam AWS Control Tower dokumentasi dan lihat Kontrol proaktif dalam Menerapkan kontrol keamanan pada AWS.

manajemen siklus hidup produk (PLM)

Manajemen data dan proses untuk suatu produk di seluruh siklus hidupnya, mulai dari desain, pengembangan, dan peluncuran, melalui pertumbuhan dan kematangan, hingga penurunan dan penghapusan.

lingkungan produksi

Lihat lingkungan.

pengontrol logika yang dapat diprogram (PLC)

Di bidang manufaktur, komputer yang sangat andal dan mudah beradaptasi yang memantau mesin dan mengotomatiskan proses manufaktur.

rantai cepat

Menggunakan output dari satu prompt LLM sebagai input untuk prompt berikutnya untuk menghasilkan respons yang lebih baik. Teknik ini digunakan untuk memecah tugas yang kompleks menjadi subtugas, atau untuk secara iteratif memperbaiki atau memperluas respons awal. Ini membantu meningkatkan akurasi dan relevansi respons model dan memungkinkan hasil yang lebih terperinci dan dipersonalisasi.

pseudonimisasi

Proses penggantian pengidentifikasi pribadi dalam kumpulan data dengan nilai placeholder. Pseudonimisasi dapat membantu melindungi privasi pribadi. Data pseudonim masih dianggap sebagai data pribadi.

publish/subscribe (pub/sub)

Pola yang memungkinkan komunikasi asinkron antara layanan mikro untuk meningkatkan skalabilitas dan daya tanggap. Misalnya, dalam MES berbasis layanan mikro, layanan mikro dapat mempublikasikan pesan peristiwa ke saluran yang dapat berlangganan layanan mikro lainnya. Sistem dapat menambahkan layanan mikro baru tanpa mengubah layanan penerbitan.

Q

rencana kueri

Serangkaian langkah, seperti instruksi, yang digunakan untuk mengakses data dalam sistem database relasional SQL.

regresi rencana kueri

Ketika pengoptimal layanan database memilih rencana yang kurang optimal daripada sebelum perubahan yang diberikan ke lingkungan database. Hal ini dapat disebabkan oleh perubahan statistik, kendala, pengaturan lingkungan, pengikatan parameter kueri, dan pembaruan ke mesin database.

R

Matriks RACI

Lihat bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan (RACI).

LAP

Lihat Retrieval Augmented Generation.

ransomware

Perangkat lunak berbahaya yang dirancang untuk memblokir akses ke sistem komputer atau data sampai pembayaran dilakukan.

Matriks RASCI

Lihat bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan (RACI).

RCAC

Lihat kontrol akses baris dan kolom.

replika baca

Salinan database yang digunakan untuk tujuan read-only. Anda dapat merutekan kueri ke replika baca untuk mengurangi beban pada database utama Anda.

arsitek ulang

Lihat 7 Rs.

tujuan titik pemulihan (RPO)

Jumlah waktu maksimum yang dapat diterima sejak titik pemulihan data terakhir. Ini menentukan apa yang dianggap sebagai kehilangan data yang dapat diterima antara titik pemulihan terakhir dan gangguan layanan.

tujuan waktu pemulihan (RTO)

Penundaan maksimum yang dapat diterima antara gangguan layanan dan pemulihan layanan.

refactor

Lihat 7 Rs.

Region

Kumpulan AWS sumber daya di wilayah geografis. Masing-masing AWS Region terisolasi dan independen dari yang lain untuk memberikan toleransi kesalahan, stabilitas, dan ketahanan. Untuk informasi selengkapnya, lihat Menentukan Wilayah AWS akun yang dapat digunakan.

regresi

Teknik ML yang memprediksi nilai numerik. Misalnya, untuk memecahkan masalah “Berapa harga rumah ini akan dijual?” Model ML dapat menggunakan model regresi linier untuk memprediksi harga jual rumah berdasarkan fakta yang diketahui tentang rumah (misalnya, luas persegi).

rehost

Lihat 7 Rs.

melepaskan

Dalam proses penyebaran, tindakan mempromosikan perubahan pada lingkungan produksi.

memindahkan

Lihat 7 Rs.

memplatform ulang

Lihat 7 Rs.

pembelian kembali

Lihat 7 Rs.

ketahanan

Kemampuan aplikasi untuk melawan atau pulih dari gangguan. Ketersediaan tinggi dan pemulihan bencana adalah pertimbangan umum ketika merencanakan ketahanan di. AWS Cloud Untuk informasi lebih lanjut, lihat AWS Cloud Ketahanan.

kebijakan berbasis sumber daya

Kebijakan yang dilampirkan ke sumber daya, seperti bucket Amazon S3, titik akhir, atau kunci enkripsi. Jenis kebijakan ini menentukan prinsipal mana yang diizinkan mengakses, tindakan yang didukung, dan kondisi lain yang harus dipenuhi.

matriks yang bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan (RACI)

Matriks yang mendefinisikan peran dan tanggung jawab untuk semua pihak yang terlibat dalam kegiatan migrasi dan operasi cloud. Nama matriks berasal dari jenis tanggung jawab yang didefinisikan dalam matriks: bertanggung jawab (R), akuntabel (A), dikonsultasikan (C), dan diinformasikan (I). Tipe dukungan (S) adalah opsional. Jika Anda menyertakan dukungan, matriks disebut matriks RASCI, dan jika Anda mengecualikannya, itu disebut matriks RACI.

kontrol responsif

Kontrol keamanan yang dirancang untuk mendorong remediasi efek samping atau penyimpangan dari garis dasar keamanan Anda. Untuk informasi selengkapnya, lihat Kontrol responsif dalam Menerapkan kontrol keamanan pada AWS.

melestarikan

Lihat 7 Rs.

pensiun

Lihat 7 Rs.

Retrieval Augmented Generation (RAG)

Teknologi AI generatif di mana LLM merujuk sumber data otoritatif yang berada di luar sumber data pelatihannya sebelum menghasilkan respons. Misalnya, model RAG mungkin melakukan

pencarian semantik dari basis pengetahuan organisasi atau data kustom. Untuk informasi lebih lanjut, lihat Apa itu RAG.

rotasi

Proses memperbarui rahasia secara berkala untuk membuatnya lebih sulit bagi penyerang untuk mengakses kredensil.

kontrol akses baris dan kolom (RCAC)

Penggunaan ekspresi SQL dasar dan fleksibel yang telah menetapkan aturan akses. RCAC terdiri dari izin baris dan topeng kolom.

RPO

Lihat tujuan titik pemulihan.

RTO

Lihat tujuan waktu pemulihan.

buku runbook

Satu set prosedur manual atau otomatis yang diperlukan untuk melakukan tugas tertentu. Ini biasanya dibangun untuk merampingkan operasi berulang atau prosedur dengan tingkat kesalahan yang tinggi.

D

SAML 2.0

Standar terbuka yang digunakan oleh banyak penyedia identitas (IdPs). Fitur ini memungkinkan sistem masuk tunggal gabungan (SSO), sehingga pengguna dapat masuk ke Konsol Manajemen AWS atau memanggil operasi AWS API tanpa Anda harus membuat pengguna di IAM untuk semua orang di organisasi Anda. Untuk informasi lebih lanjut tentang federasi berbasis SAMP 2.0, lihat Tentang federasi berbasis SAMP 2.0 dalam dokumentasi IAM.

SCADA

Lihat kontrol pengawasan dan akuisisi data.

SCP

Lihat kebijakan kontrol layanan.

Rahasia

Dalam AWS Secrets Manager, informasi rahasia atau terbatas, seperti kata sandi atau kredensil pengguna, yang Anda simpan dalam bentuk terenkripsi. Ini terdiri dari nilai rahasia dan metadata-nya. Nilai rahasia dapat berupa biner, string tunggal, atau beberapa string. Untuk informasi selengkapnya, lihat Apa yang ada di rahasia Secrets Manager? dalam dokumentasi Secrets Manager.

keamanan dengan desain

Pendekatan rekayasa sistem yang memperhitungkan keamanan melalui seluruh proses pengembangan.

kontrol keamanan

Pagar pembatas teknis atau administratif yang mencegah, mendeteksi, atau mengurangi kemampuan pelaku ancaman untuk mengeksploitasi kerentanan keamanan. Ada empat jenis kontrol keamanan utama: preventif, detektif, responsif, dan proaktif.

pengerasan keamanan

Proses mengurangi permukaan serangan untuk membuatnya lebih tahan terhadap serangan. Ini dapat mencakup tindakan seperti menghapus sumber daya yang tidak lagi diperlukan, menerapkan praktik keamanan terbaik untuk memberikan hak istimewa paling sedikit, atau menonaktifkan fitur yang tidak perlu dalam file konfigurasi.

sistem informasi keamanan dan manajemen acara (SIEM)

Alat dan layanan yang menggabungkan sistem manajemen informasi keamanan (SIM) dan manajemen acara keamanan (SEM). Sistem SIEM mengumpulkan, memantau, dan menganalisis data dari server, jaringan, perangkat, dan sumber lain untuk mendeteksi ancaman dan pelanggaran keamanan, dan untuk menghasilkan peringatan.

otomatisasi respons keamanan

Tindakan yang telah ditentukan dan diprogram yang dirancang untuk secara otomatis merespons atau memulihkan peristiwa keamanan. Otomatisasi ini berfungsi sebagai kontrol keamanan detektif atau responsif yang membantu Anda menerapkan praktik terbaik AWS keamanan. Contoh tindakan respons otomatis termasuk memodifikasi grup keamanan VPC, menambal instans Amazon EC2, atau memutar kredensil.

enkripsi sisi server

Enkripsi data di tujuannya, oleh Layanan AWS yang menerimanya.

kebijakan kontrol layanan (SCP)

Kebijakan yang menyediakan kontrol terpusat atas izin untuk semua akun di organisasi. AWS Organizations SCPs menentukan pagar pembatas atau menetapkan batasan pada tindakan yang dapat didelegasikan oleh administrator kepada pengguna atau peran. Anda dapat menggunakan SCPs daftar izin atau daftar penolakan, untuk menentukan layanan atau tindakan mana yang diizinkan atau dilarang. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam AWS Organizations dokumentasi.

titik akhir layanan

URL titik masuk untuk file Layanan AWS. Anda dapat menggunakan endpoint untuk terhubung secara terprogram ke layanan target. Untuk informasi selengkapnya, lihat Layanan AWS titik akhir di Referensi Umum AWS.

perjanjian tingkat layanan (SLA)

Perjanjian yang menjelaskan apa yang dijanjikan tim TI untuk diberikan kepada pelanggan mereka, seperti waktu kerja dan kinerja layanan.

indikator tingkat layanan (SLI)

Pengukuran aspek kinerja layanan, seperti tingkat kesalahan, ketersediaan, atau throughputnya.

tujuan tingkat layanan (SLO)

Metrik target yang mewakili kesehatan layanan, yang diukur dengan indikator tingkat layanan.

model tanggung jawab bersama

Model yang menjelaskan tanggung jawab yang Anda bagikan AWS untuk keamanan dan kepatuhan cloud. AWS bertanggung jawab atas keamanan cloud, sedangkan Anda bertanggung jawab atas keamanan di cloud. Untuk informasi selengkapnya, lihat Model tanggung jawab bersama.

SIEM

Lihat informasi keamanan dan sistem manajemen acara.

titik kegagalan tunggal (SPOF)

Kegagalan dalam satu komponen penting dari aplikasi yang dapat mengganggu sistem.

SLA

Lihat perjanjian tingkat layanan.

SLI

Lihat indikator tingkat layanan.

SLO

Lihat tujuan tingkat layanan.

split-and-seed model

Pola untuk menskalakan dan mempercepat proyek modernisasi. Ketika fitur baru dan rilis produk didefinisikan, tim inti berpisah untuk membuat tim produk baru. Ini membantu meningkatkan kemampuan dan layanan organisasi Anda, meningkatkan produktivitas pengembang, dan mendukung inovasi yang cepat. Untuk informasi lebih lanjut, lihat Pendekatan bertahap untuk memodernisasi aplikasi di. AWS Cloud

SPOF

Lihat satu titik kegagalan.

skema bintang

Struktur organisasi database yang menggunakan satu tabel fakta besar untuk menyimpan data transaksional atau terukur dan menggunakan satu atau lebih tabel dimensi yang lebih kecil untuk menyimpan atribut data. Struktur ini dirancang untuk digunakan dalam gudang data atau untuk tujuan intelijen bisnis.

pola ara pencekik

Pendekatan untuk memodernisasi sistem monolitik dengan menulis ulang secara bertahap dan mengganti fungsionalitas sistem sampai sistem warisan dapat dinonaktifkan. Pola ini menggunakan analogi pohon ara yang tumbuh menjadi pohon yang sudah mapan dan akhirnya mengatasi dan menggantikan inangnya. Pola ini diperkenalkan oleh Martin Fowler sebagai cara untuk mengelola risiko saat menulis ulang sistem monolitik. Untuk contoh cara menerapkan pola ini, lihat Memodernisasi layanan web Microsoft ASP.NET (ASMX) lama secara bertahap menggunakan container dan Amazon API Gateway.

subnet

Rentang alamat IP dalam VPC Anda. Subnet harus berada di Availability Zone tunggal.

kontrol pengawasan dan akuisisi data (SCADA)

Di bidang manufaktur, sistem yang menggunakan perangkat keras dan perangkat lunak untuk memantau aset fisik dan operasi produksi.

enkripsi simetris

Algoritma enkripsi yang menggunakan kunci yang sama untuk mengenkripsi dan mendekripsi data.

pengujian sintetis

Menguji sistem dengan cara yang mensimulasikan interaksi pengguna untuk mendeteksi potensi masalah atau untuk memantau kinerja. Anda dapat menggunakan Amazon CloudWatch Synthetics untuk membuat tes ini.

sistem prompt

Teknik untuk memberikan konteks, instruksi, atau pedoman ke LLM untuk mengarahkan perilakunya. Permintaan sistem membantu mengatur konteks dan menetapkan aturan untuk interaksi dengan pengguna.

T

tag

Pasangan nilai kunci yang bertindak sebagai metadata untuk mengatur sumber daya Anda. AWS Tanda membantu Anda mengelola, mengidentifikasi, mengatur, dan memfilter sumber daya. Untuk informasi selengkapnya, lihat Menandai AWS sumber daya Anda.

variabel target

Nilai yang Anda coba prediksi dalam ML yang diawasi. Ini juga disebut sebagai variabel hasil. Misalnya, dalam pengaturan manufaktur, variabel target bisa menjadi cacat produk.

daftar tugas

Alat yang digunakan untuk melacak kemajuan melalui runbook. Daftar tugas berisi ikhtisar runbook dan daftar tugas umum yang harus diselesaikan. Untuk setiap tugas umum, itu termasuk perkiraan jumlah waktu yang dibutuhkan, pemilik, dan kemajuan.

lingkungan uji

Lihat lingkungan.

pelatihan

Untuk menyediakan data bagi model ML Anda untuk dipelajari. Data pelatihan harus berisi jawaban yang benar. Algoritma pembelajaran menemukan pola dalam data pelatihan yang

memetakan atribut data input ke target (jawaban yang ingin Anda prediksi). Ini menghasilkan model ML yang menangkap pola-pola ini. Anda kemudian dapat menggunakan model ML untuk membuat prediksi pada data baru yang Anda tidak tahu targetnya.

gerbang transit

Hub transit jaringan yang dapat Anda gunakan untuk menghubungkan jaringan Anda VPCs dan lokal. Untuk informasi selengkapnya, lihat Apa itu gateway transit dalam AWS Transit Gateway dokumentasi.

alur kerja berbasis batang

Pendekatan di mana pengembang membangun dan menguji fitur secara lokal di cabang fitur dan kemudian menggabungkan perubahan tersebut ke cabang utama. Cabang utama kemudian dibangun untuk pengembangan, praproduksi, dan lingkungan produksi, secara berurutan.

akses tepercaya

Memberikan izin ke layanan yang Anda tentukan untuk melakukan tugas di organisasi Anda di dalam AWS Organizations dan di akunnya atas nama Anda. Layanan tepercaya menciptakan peran terkait layanan di setiap akun, ketika peran itu diperlukan, untuk melakukan tugas manajemen untuk Anda. Untuk informasi selengkapnya, lihat Menggunakan AWS Organizations dengan AWS layanan lain dalam AWS Organizations dokumentasi.

penyetelan

Untuk mengubah aspek proses pelatihan Anda untuk meningkatkan akurasi model ML. Misalnya, Anda dapat melatih model ML dengan membuat set pelabelan, menambahkan label, dan kemudian mengulangi langkah-langkah ini beberapa kali di bawah pengaturan yang berbeda untuk mengoptimalkan model.

tim dua pizza

Sebuah DevOps tim kecil yang bisa Anda beri makan dengan dua pizza. Ukuran tim dua pizza memastikan peluang terbaik untuk berkolaborasi dalam pengembangan perangkat lunak.

U

waswas

Sebuah konsep yang mengacu pada informasi yang tidak tepat, tidak lengkap, atau tidak diketahui yang dapat merusak keandalan model ML prediktif. Ada dua jenis ketidakpastian: ketidakpastian epistemik disebabkan oleh data yang terbatas dan tidak lengkap, sedangkan

ketidakpastian aleatorik disebabkan oleh kebisingan dan keacakan yang melekat dalam data. Untuk informasi lebih lanjut, lihat panduan Mengukur ketidakpastian dalam sistem pembelajaran mendalam.

tugas yang tidak terdiferensiasi

Juga dikenal sebagai angkat berat, pekerjaan yang diperlukan untuk membuat dan mengoperasikan aplikasi tetapi itu tidak memberikan nilai langsung kepada pengguna akhir atau memberikan keunggulan kompetitif. Contoh tugas yang tidak terdiferensiasi termasuk pengadaan, pemeliharaan, dan perencanaan kapasitas.

lingkungan atas

Lihat lingkungan.

V

menyedot debu

Operasi pemeliharaan database yang melibatkan pembersihan setelah pembaruan tambahan untuk merebut kembali penyimpanan dan meningkatkan kinerja.

kendali versi

Proses dan alat yang melacak perubahan, seperti perubahan kode sumber dalam repositori.

Peering VPC

Koneksi antara dua VPCs yang memungkinkan Anda untuk merutekan lalu lintas dengan menggunakan alamat IP pribadi. Untuk informasi selengkapnya, lihat Apa itu peering VPC di dokumentasi VPC Amazon.

kerentanan

Kelemahan perangkat lunak atau perangkat keras yang membahayakan keamanan sistem.

W

cache hangat

Cache buffer yang berisi data terkini dan relevan yang sering diakses. Instance database dapat membaca dari cache buffer, yang lebih cepat daripada membaca dari memori utama atau disk.

data hangat

Data yang jarang diakses. Saat menanyakan jenis data ini, kueri yang cukup lambat biasanya dapat diterima.

fungsi jendela

Fungsi SQL yang melakukan perhitungan pada sekelompok baris yang berhubungan dengan catatan saat ini. Fungsi jendela berguna untuk memproses tugas, seperti menghitung rata-rata bergerak atau mengakses nilai baris berdasarkan posisi relatif dari baris saat ini.

beban kerja

Kumpulan sumber daya dan kode yang memberikan nilai bisnis, seperti aplikasi yang dihadapi pelanggan atau proses backend.

aliran kerja

Grup fungsional dalam proyek migrasi yang bertanggung jawab atas serangkaian tugas tertentu. Setiap alur kerja independen tetapi mendukung alur kerja lain dalam proyek. Misalnya, alur kerja portofolio bertanggung jawab untuk memprioritaskan aplikasi, perencanaan gelombang, dan mengumpulkan metadata migrasi. Alur kerja portofolio mengirimkan aset ini ke alur kerja migrasi, yang kemudian memigrasikan server dan aplikasi.

CACING

Lihat menulis sekali, baca banyak.

WQF

Lihat AWS Kerangka Kualifikasi Beban Kerja.

tulis sekali, baca banyak (WORM)

Model penyimpanan yang menulis data satu kali dan mencegah data dihapus atau dimodifikasi. Pengguna yang berwenang dapat membaca data sebanyak yang diperlukan, tetapi mereka tidak dapat mengubahnya. Infrastruktur penyimpanan data ini dianggap tidak dapat diubah.

Z

eksploitasi zero-day

Serangan, biasanya malware, yang memanfaatkan kerentanan zero-day.

kerentanan zero-day

Cacat atau kerentanan yang tak tanggung-tanggung dalam sistem produksi. Aktor ancaman dapat menggunakan jenis kerentanan ini untuk menyerang sistem. Pengembang sering menyadari kerentanan sebagai akibat dari serangan tersebut.

bisikan zero-shot

Memberikan LLM dengan instruksi untuk melakukan tugas tetapi tidak ada contoh (tembakan) yang dapat membantu membimbingnya. LLM harus menggunakan pengetahuan pra-terlatih untuk menangani tugas. Efektivitas bidikan nol tergantung pada kompleksitas tugas dan kualitas prompt. Lihat juga beberapa bidikan yang diminta.

aplikasi zombie

Aplikasi yang memiliki CPU rata-rata dan penggunaan memori di bawah 5 persen. Dalam proyek migrasi, adalah umum untuk menghentikan aplikasi ini.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.