Engineer Daten

Datenbanken

30 Leistungen

Die Teile Ihres Systems verbinden

Die unter „Datenbanken“ gesammelten Arbeiten zeigen einen Teil eines funktionierenden Systems. Wir können helfen, diesen Teil mit dem Rest Ihres Betriebs zu verbinden. Die Abfragen und fehlenden Indizes hinter einem von Jahr zu Jahr schwerfälligeren System finden und die entscheidenden Probleme beheben.

Unsere Leistungen entdecken Kontakt aufnehmen →

Eine vergleichbare Herausforderung finden

Folgen Sie einem Beispiel, das Ihrer Situation ähnelt. Die Rückschau beschreibt die Entscheidungen hinter dem Ergebnis und geht über die bloße Technologiebezeichnung hinaus. 30 Tage Stromdaten in 180 ms — 180 ms für die Aggregation eines Monats an Zählerständen

30 Tage Stromdaten in 180 ms Kontakt aufnehmen →

30 Tage Stromdaten in 180 ms

Varianten zur stündlichen Aggregation des Stromverbrauchs in Python, SQL und Bash + Jq entwickelt; der Python-Benchmark verarbeitete einen 30-Tage-JSONL-Datensatz in 180 ms.

Geografische Datenpipeline 50-mal schneller

Eine 50-fache Verbesserung der geografischen Datenpipeline bei der Arbeit der Gründerin für Utiligize durch Änderungen an SQL und Datenmodellen in PostgreSQL, MS SQL und Google Cloud BigQuery dokumentiert.

GIS-Karte mit PostgreSQL 10-mal schneller

Die Performance einer geografischen Kartenanwendung durch einen gezielten Datenbankwechsel von MSSQL zu PostgreSQL einschließlich Datenumschaltung um das 10-Fache verbessert und dabei Verarbeitung und Datensicherheit optimiert.

1.000 Fehler in geografischen Daten behoben

1.000 Probleme in geografischen Daten und Zeitreihendaten mit GDAL, ArcGIS, PostGIS, Mapbox, QGIS, SQL (PL/pgSQL, Transact-SQL) und Bash behoben und eine hochwertige Big-Data-Verarbeitung sichergestellt.

6 ETL/ELT-Pipelines entworfen und betrieben

6 ETL/ELT-Pipelines mit Google BigQuery, MSSQL, PostgreSQL, Shell-Skripten, PL/pgSQL und Transact-SQL entworfen, umgesetzt und administriert sowie Daten für eine effiziente Verarbeitung durch die Python-API integriert.

500 Strom- und GIS-Datenberichte veröffentlicht

500 Analyseberichte zu Strom- und GIS-Daten veröffentlicht und durch gründliche Recherche und Fehlerdiagnose präzise Erkenntnisse aus geografischen Big Data und Zeitreihendaten ermöglicht.

100 PostgreSQL/MS-SQL/BigQuery-Warehouses

100 Data-Warehouse-Datenbanken in PostgreSQL, MS SQL und Google BigQuery mit überwiegend GIS- und Zeitreihendaten konzipiert, aufgebaut und verwaltet sowie Performance und Skalierbarkeit optimiert.

10 PostgreSQL-/MS-SQL-Server auf Ubuntu

10 PostgreSQL- und MS-SQL-Server auf Ubuntu-Linux-VPS entworfen, bereitgestellt und gewartet und dabei Serverperformance und Zuverlässigkeit sichergestellt.

1.000 RBAC-Regeln für sichere Linux-Server

Die Datensicherheit durch 1.000 RBAC-Regeln für Entwickler, Anwendungsinstanzen, PostgreSQL, MS SQL und weitere Linux-Server erhöht, unberechtigte Zugriffe verhindert und die Regeln mit Ansible-Automatisierung dokumentiert.

PostgreSQL um das 10-Fache beschleunigt

PostgreSQL durch gezielte Indizierung, Partitionierung und Abfrageoptimierung um das 10-Fache beschleunigt und die Datenbankeffizienz für Nutzer-, Mandanten-, Geodaten und Strom-Zeitreihendaten verbessert.

100 wichtige Sicherungen mit Barman automatisiert

100 geschäftskritische Datensicherungen mit Barman, Google Cloud, Bash und Python automatisiert und die Datenintegrität über die Datenbanken hinweg abgesichert.

30+ GIS-Projekte von Entwicklung bis Support

Entwicklung, Bereitstellung und Support von über 30 GIS-Projekten geleitet und dabei Fachwissen zu PostgreSQL, Bash, Python, JavaScript, GDAL, ArcGIS, PostGIS und Mapbox eingesetzt.

Datenarbeit mit Python und PL/pgSQL automatisiert

Datenverarbeitungsaufgaben mit Shell-Skripten, PL/pgSQL, Python und Transact-SQL automatisiert und Produktivität sowie Effizienz gesteigert.

600 PL/pgSQL-ETL/ELT-Pipelines entwickelt

600 PL/pgSQL-basierte ETL/ELT-Pipelines für effizientere komplexe Datenverarbeitung in mehreren PostgreSQL-Entwicklungs- und Produktivumgebungen entwickelt.

Eine maritime Plattform mit klaren Systemgrenzen

Eine geschichtete maritime Plattform mit getrenntem Next.js-PWA-Frontend, Go-API (Huma/Fiber) und PostgreSQL-Funktionsschicht entworfen und sämtliche Geschäftslogik in der Datenbank gehalten.

JSON direkt von Postgres an die API geleitet

Eine JSON-Passthrough-Architektur entworfen, in der PostgreSQL-Funktionen vollständiges JSON zurückgeben, das die Go-API unverändert weiterreicht; damit entfällt die zwischenzeitliche Deserialisierung und das Frontend bleibt von Schemaänderungen entkoppelt.

PostgreSQL-Datenschicht mit Funktionen entworfen

Eine PostgreSQL-Datenschicht nach dem Function-first-Prinzip entworfen — 1.275 gespeicherte Funktionen in 34 Schemata — sodass jeder Lese- und Schreibzugriff über eine gezielt berechtigbare Funktion statt über eine Tabelle erfolgt.

Zeitlich geordnete Schlüssel für ein wachsendes Datenmodell

Zeitlich geordnete UUID-v7-Identifikatoren (PostgreSQL 18) als Entitätsschlüssel eingeführt, um die Fragmentierung von B-Baum-Indizes zu verringern und Abfragen zu beschleunigen.

Kataloggesteuerten Deep-Merge umgesetzt

Eine kataloggesteuerte tiefe Zusammenführung gespeicherter JSON-Einstellungen umgesetzt, die Abstürze durch fehlende Schlüssel bei der Weiterentwicklung des Schemas verhindert.

348 Tabellen in 34 Postgres-Schemata modelliert

Die maritime Domäne in 348 normalisierten Tabellen über 34 PostgreSQL-Schemata modelliert — Fachkräfte, Unternehmen, Schiffe, Stellen, Bewertungen und weitere Bereiche — mit SMALLINT-Nachschlagetabellen und UUID-v7-Schlüsseln.

Sicherungen und Notfallwiederherstellung als Code

Die Aufbewahrung von Datenbanksicherungen als Infrastructure as Code konfiguriert, anschließend die Wiederherstellbarkeit geprüft und das Wiederherstellungsverfahren dokumentiert — mit ausdrücklich benannten Lücken, statt sie erst im Ernstfall entdecken zu lassen.

Stripe- und In-App-Kaufberechtigungen entwickelt

Die Zahlungs- und Berechtigungsschicht aufgebaut — Stripe sowie In-App-Käufe von Apple und Google — und Verzeichnis, Suche und Export durch ein serverseitig geprüftes Zugriffsmodell mit 11 Tabellen abgesichert.

Langsame Aufgaben in River-Jobqueue verlagert

Langsame Verarbeitung aus dem Anfragepfad in eine River-Jobqueue verlagert — 15 Worker-Module, 8 geplante Aufgaben und 20 pg_cron-Jobs — sodass Anfragen zurückkehren, während die zugehörige Arbeit weiterläuft.

1.022 Migrationen mit Schemaabweichungsprüfung

Die Konsistenz des Schemas über 1.022 Migrationen mit einer CI-Prüfung abgesichert, die die Datenbank auf zwei Wegen aufbaut — Neuinstallation sowie Installation mit sämtlichen Migrationen — und bei Unterschieden fehlschlägt.

Eigene Analysen in PostgreSQL entwickelt

Eigene Produktanalysen in PostgreSQL entwickelt — 47 Funktionen über partitionierten Ereignistabellen mit automatischer Bereinigung — pseudonymisiert mit wechselndem Salt und nur mit Einwilligung des Besuchers.

Loyalitätskonto, Ligen und Abzeichen entwickelt

Das Loyalitäts- und Reputationssystem entwickelt — 67 Funktionen über ein Hauptbuch mit 31 Tabellen, mit Ligen, Abzeichen und einem Prämienshop — und den Kontostand durch eine Zeilensperre gegen doppelte Ausgaben abgesichert.

Stellenmarkt und Seefahrtskarriere entwickelt

Arbeitsbereiche für Personalgewinnung und die Laufbahn von Seeleuten entwickelt; der dokumentierte Implementierungsstand umfasst 151 gespeicherte Funktionen in 48 Tabellen für Stellen, Bewerbungen, Zertifikate, Rangentwicklung und Seefahrtzeit.

495 Seiten aus einem SQLite-Export erzeugt

495 Leistungsseiten in drei Sprachen aus einem schreibgeschützten SQLite-Export erzeugt und die Seitenadresse als eigene Daten festgehalten, damit eine Satzkorrektur nicht länger die Seite verschiebt und Links bricht.

Datenbankgestützten Lebenslaufgenerator aufgebaut

Einen datenbankgestützten Generator für Lebenslauf, Referenzen, Portfolio und Anschreiben in Python entwickelt — 41 Module, 10.580 Zeilen — der sechs Ausgabeformate aus einer SQLite-Quelle mit 23 Tabellen erzeugt, aufgebaut durch eine idempotente Pipeline mit 19 Schritten.

Die Prüfwerkzeuge selbst mit Tests geprüft

Tests für die Prüfwerkzeuge selbst geschrieben, nachdem klar war, dass ein nur mit fehlerfreien Eingaben getestetes Werkzeug irgendwann auch nach dem Lesen von nichts Erfolg meldet — einen Tippfehler eingebaut, um die Rechtschreibprüfung zu prüfen, und einen ID-Bereich aus der Datenbank statt einer Testkonstante übernommen.

Die technische Arbeit gezielt einsetzen

Bringen Sie einen repräsentativen Datensatz, die Fragen, die er beantworten muss, und den aktuellen Engpass mit. Wir können vereinbaren, wie Qualität, Durchsatz und Wiederherstellung gemessen werden.

Preise ansehen Kontakt aufnehmen →

Das Thema in Ihren Kontext einordnen

Erzählen Sie uns, welche Rolle „Datenbanken“ in Ihrem eigenen System spielt. Wir können helfen, die passende Leistung auszuwählen und die damit verbundene Arbeit zu definieren.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://data.engineer.company/de/categories/databases/