Engineer

Databaser

35 resultater

Knytt sammen delene av systemet

Arbeidet under «Databaser» viser én del av et fungerende system. Vi kan hjelpe med å knytte den delen til resten av driften deres. Finne spørringene og de manglende indeksene bak et system som har blitt tyngre for hvert år, og rette dem som betyr noe.

Se tjenestene våre Ta kontakt →

Finn en sammenlignbar utfordring

Åpne et eksempel som ligner situasjonen deres. Gjennomgangen forklarer beslutningene bak resultatet, utover navnet på teknologien. 30 dagers strømdata på 180 ms — 180 ms for å aggregere en måned med måleravlesninger

30 dagers strømdata på 180 ms Ta kontakt →

30 dagers strømdata på 180 ms

Utviklet varianter for timesaggregering av strømforbruk i Python, SQL og Bash + Jq; Python-varianten behandlet ett JSONL-datasett for 30 dager på 180 ms i ytelsestesten.

Gjorde en geodatapipeline 50 ganger raskere

Registrerte en forbedring på 50 ganger i en geografisk datapipeline i grunnleggerens arbeid hos Utiligize gjennom SQL- og datamodellendringer i PostgreSQL, MS SQL og Google Cloud BigQuery.

Gjorde GIS-kart 10 ganger raskere på PostgreSQL

Forbedret ytelsen i en geografisk kartapplikasjon 10 ganger gjennom et strategisk databaseskifte fra MSSQL til PostgreSQL, inkludert omleggingen av dataene, og optimaliserte samtidig behandlingen og datasikkerheten.

Løste 1 000 problemer i geografiske data

Løste 1 000 problemer i geografiske data og tidsseriedata med GDAL, ArcGIS, PostGIS, Mapbox, QGIS, SQL (PL/pgSQL, Transact-SQL) og Bash, og sikret behandling av stordata med høy kvalitet.

Designet og driftet 6 ETL/ELT-pipelines

Designet, implementerte og administrerte 6 ETL/ELT-pipelines med Google BigQuery, MSSQL, PostgreSQL, Shell-skripting, PL/pgSQL og Transact-SQL, og integrerte data for effektiv behandling i et Python-API.

Publiserte 500 datarapporter om strøm og GIS

Publiserte 500 analyserapporter om strøm- og GIS-data, basert på grundige undersøkelser og feilsøking, for å sikre presis innsikt i geografiske stordata og tidsseriedata.

Drev 100 datavarehus: Postgres, MS SQL, BigQuery

Utformet, opprettet og administrerte 100 databaser for datavarehus i PostgreSQL, MS SQL og Google BigQuery, hovedsakelig med GIS- og tidsseriedata, og optimaliserte ytelse og skalerbarhet.

Drev 10 PostgreSQL- og MS SQL-servere på Ubuntu

Designet, satte i produksjon og vedlikeholdt 10 PostgreSQL- og MS SQL-servere på Ubuntu Linux-VPS-er, og sikret optimal serverytelse og pålitelighet.

Skrev 1 000 RBAC-regler for å sikre Linux-servere

Styrket datasikkerheten ved å innføre 1 000 RBAC-regler for utviklere, applikasjonsinstanser, PostgreSQL, MS SQL og andre Linux-servere, og forhindret uautorisert tilgang; dokumentert med Ansible-automatisering.

Økte PostgreSQL-ytelsen 10 ganger

Økte PostgreSQL-ytelsen 10 ganger gjennom strategisk indeksering, partisjonering og spørringsoptimalisering, og gjorde databasen mer effektiv for bruker- og tenantdata, geodata og elektriske tidsseriedata.

Automatiserte 100 kritiske sikkerhetskopier

Automatiserte 100 kritiske sikkerhetskopier av data med Barman, Google Cloud, Bash og Python, og sikret dataintegriteten på tvers av databasene.

Styrte fullstack-utviklingen av GIS-kart

Styrte fullstack-utviklingen av GIS-kart med ansvar for PostgreSQL, Mapbox, ReactJS og NodeJS, og leverte en integrert løsning.

Ledet 30+ GIS-prosjekter fra utvikling til støtte

Ledet utvikling, produksjonssetting og støtte for over 30 GIS-prosjekter, med ekspertise i teknologiene PostgreSQL, Bash, Python, JavaScript, GDAL, ArcGIS, PostGIS og Mapbox.

Automatiserte databehandling i Python og PL/pgSQL

Automatiserte databehandlingsoppgaver med Shell-skripting, PL/pgSQL, Python og Transact-SQL, og økte produktiviteten og effektiviteten.

Utviklet 600 ETL/ELT-pipelines i PL/pgSQL

Utviklet 600 PL/pgSQL-baserte ETL/ELT-pipelines for å effektivisere komplekse arbeidsflyter for databehandling på tvers av flere utviklings- og produksjonsmiljøer i PostgreSQL.

En maritim plattform med klare systemgrenser

Utformet arkitekturen for en lagdelt maritim plattform som skiller en Next.js PWA-frontend, et Go-API (Huma/Fiber) og et funksjonslag i PostgreSQL, med all forretningslogikk i databasen.

Designet JSON rett fra Postgres til API-et

Designet en JSON-passthrough-arkitektur der PostgreSQL-funksjoner returnerer komplett JSON som Go-API-et videresender uendret, noe som fjernet mellomliggende unmarshalling og frikoblet frontend fra skjemaendringer.

Designet et funksjonsbasert datalag i PostgreSQL

Designet et funksjonsbasert datalag i PostgreSQL — 1 275 lagrede funksjoner fordelt på 34 skjemaer — slik at hver lesing og skriving går gjennom en funksjon databasen kan gi tilgang til, i stedet for gjennom en tabell.

Tidsordnede nøkler for en voksende datamodell

Tok i bruk tidsordnede UUID v7-identifikatorer (PostgreSQL 18) som entitetsnøkler for å redusere fragmenteringen i B-tre-indekser og gjøre spørringene raskere.

Implementerte en katalogstyrt deep merge

Implementerte en katalogstyrt deep merge av lagrede JSON-innstillinger, som forhindrer krasj ved manglende nøkler når skjemaet utvikler seg.

Modellerte 348 tabeller i 34 Postgres-skjemaer

Modellerte det maritime domenet i 348 normaliserte tabeller fordelt på 34 PostgreSQL-skjemaer — fagfolk, selskaper, skip, stillinger, omtaler og resten — med SMALLINT-oppslag og UUID v7-nøkler.

La sikkerhetskopi og katastrofeberedskap i kode

Konfigurerte lagringstiden for databasens sikkerhetskopier som infrastruktur som kode, gjennomgikk deretter gjenopprettingsberedskapen og dokumenterte fremgangsmåten for gjenoppretting — og navnga de gjenværende hullene i stedet for å la dem bli oppdaget under en hendelse.

Herdet applikasjonen med nonce-basert CSP

Herdet applikasjonen med nonce-basert CSP, HSTS, SameSite-informasjonskapsler, databaseroller etter prinsippet om minste privilegium og ny kontroll av tilgangsrettigheter på serversiden.

Tok plattformens grunnleggende tekniske valg

Tok plattformens grunnleggende beslutninger i ukene etter at repoet ble opprettet i november 2025 — lagdelingen, datatilgang med databasen først og kravet om null advarsler — og de står seg fortsatt ni måneder senere.

Bygde tilgangsrettigheter for Stripe og appkjøp

Bygde laget for betaling og tilgangsrettigheter — Stripe sammen med kjøp i appen hos Apple og Google — som styrer tilgangen til katalogen, søket og eksporten gjennom en tilgangsmodell på 11 tabeller som kontrolleres på serveren.

Flyttet tregt arbeid over i en River-jobbkø

Flyttet tregt arbeid ut av forespørselsbanen og over i en River-jobbkø — 15 worker-moduler, 8 planlagte oppgaver og 20 pg_cron-jobber — slik at en forespørsel returnerer mens arbeidet bak den fortsetter.

1 022 migreringer bak en port for skjemaavvik

Holdt skjemaet ærlig gjennom 1 022 migreringer med en CI-port som bygger databasen begge veier — en ny installasjon, og en installasjon pluss alle migreringer — og feiler når de to ikke stemmer overens.

Bygde egen analyse inne i PostgreSQL

Bygde egen produktanalyse i PostgreSQL — 47 funksjoner over partisjonerte hendelsestabeller som rydder seg selv — pseudonymisert bak et roterende salt og betinget av den besøkendes samtykke.

Bygde en lojalitetslommebok, ligaer og merker

Bygde lojalitets- og omdømmesystemet — 67 funksjoner over en hovedbok på 31 tabeller, med ligaer, merker og en innløsningsbutikk — med radlås på saldoen for å lukke vinduet for dobbeltforbruk.

Bygde det sosiale laget: innlegg, feed og grupper

Bygde plattformens sosiale lag — innlegg, feed, grupper, nevninger, en følgergraf og et sammendrag av merkedager — på det samme funksjonsbaserte datalaget som resten av produktet.

Bygde arbeidsflater for rekruttering og sjøfolk

Bygde arbeidsflater for rekruttering og sjøfolks karriere; det registrerte øyeblikksbildet av implementasjonen omfatter 151 lagrede funksjoner i 48 tabeller for stillinger, søknader, sertifikater, rangutvikling og fartstid.

Genererte 495 sider fra en SQLite-eksport

Genererte 495 resultatsider på tre språk fra en skrivebeskyttet SQLite-eksport, med sideadressen skrevet som data, slik at en rettet setning ikke lenger flyttet siden og brøt lenken.

Bygde en databasedrevet CV-generator

Bygde en databasedrevet generator for CV, referanser, portefølje og søknadsbrev i Python — 41 moduler, 10 580 linjer — som gjengir seks utdataformater fra én SQLite-kilde med 23 tabeller, satt sammen av en idempotent pipeline i 19 trinn.

Holdt generatoren til 981 tester og 92 % dekning

Holdt generatoren til 981 testtilfeller med et minstekrav på 92 % grendekning og advarsler behandlet som feil, og verifiserte idempotens ved å kjøre hele byggepipelinen to ganger fra en tom fil og kreve at den andre kjøringen ikke endret noe.

Skrev tester for selve kontrollverktøyene

Skrev tester for selve kontrollene etter å ha slått fast at en kontroll som bare får ren inndata, en dag vil melde rent fordi den ikke leste noe — plantet en stavefeil for å bekrefte at stavekontrollen finner den, og hentet et id-område fra databasen i stedet for fra et tall i testen.

Fokuser den tekniske innsatsen

Ta med et representativt datasett, spørsmålene det skal besvare og dagens flaskehals. Vi kan avtale målinger av kvalitet, gjennomstrømning og gjenoppretting.

Se hva det koster Ta kontakt →

Sett emnet i deres sammenheng

Fortell hvordan «Databaser» inngår i systemet deres. Vi kan hjelpe med å velge riktig tjeneste og avgrense arbeidet.

Se tjenestene våre

welcome@engineer.company +45 50 10 61 84 København, Danmark

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/categories/databases/