Engineer

PostgreSQL

🐘 databasen det meste af dette arbejde er bygget på

Hvad det er

PostgreSQL er en relationel databaseserver, gratis og open source, der gemmer data i tabeller, besvarer forespørgsler skrevet i SQL og gør hver transaktion til alt eller intet. De fleste, der arbejder med den, kalder den Postgres, og projektet regner begge navne for sine egne. Bliver en erfaren udvikler spurgt »hvad er PostgreSQL?«, er svaret ligetil: det er den alsidige database, et nyt system starter på, når ingen har en konkret grund til at vælge en anden.

Den er ældre end det meste af den software, der i dag kører oven på den. POSTGRES-projektet begyndte på University of California i Berkeley under professor Michael Stonebraker, og implementeringen gik i gang i 1986. I 1994 tilføjede Andrew Yu og Jolly Chen en SQL-fortolker og udgav resultatet som Postgres95. To år senere skiftede det navn til PostgreSQL, og versionsnummereringen startede ved 6.0 i forlængelse af den række, Berkeley-projektet havde begyndt. Projektet beskriver selv systemet som objektrelationelt: ud over tabeller og joins kan en bruger tilføje nye datatyper, funktioner, operatorer, aggregatfunktioner, indeksmetoder og procedureprog, og serveren behandler dem, som var de dens egne.

Arkitekturen er klient og server. Én serverproces ejer datafilerne, og hver klientforbindelse får sin egen proces, som serveren forker til netop den forbindelse. Applikationer sender SQL over forbindelsen gennem en driver til det sprog, de er skrevet i. Samtidighed bygger på multiversion concurrency control, MVCC: en opdatering skriver en ny version af rækken i stedet for at overskrive den gamle, så hver sætning læser et konsistent øjebliksbillede, læsere venter aldrig på skrivere, og skrivere venter aldrig på læsere.

Hvor står den i forhold til alternativerne? Ved siden af MySQL og MariaDB som de andre relationelle open source-servere, ved siden af Oracle og Microsoft SQL Server som de kommercielle, ved siden af SQLite som den indlejrede database, der lever inde i ét program, og over for dokumentdatabaser som MongoDB, der gemmer poster uden et fast skema.

Brug begrebet i en beslutning

Brug »Hvad er PostgreSQL?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. At finde de forespørgsler og manglende indeks bag et system, der er blevet tungere hvert år, og rette dem, der betyder noget.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. 30 dages eldata på 180 ms — 180 ms til at aggregere en måneds måleraflæsninger

30 dages eldata på 180 ms Kontakt os →

Hvorfor det betyder noget

Valget af database er en beslutning, et team lever med i årevis, fordi data overlever den kode, der er skrevet mod dem. PostgreSQL er det standardvalg, man sjældent fortryder, og det af tre grunde, som en køber selv kan efterprøve.

Den første er pris og leverandørbinding. Den er udgivet under PostgreSQL-licensen, en kort og liberal licens i samme familie som BSD og MIT, så der er ingen betaling pr. kerne, ingen licensrevision og ingen leverandør, der kan ændre vilkårene. Amazon, Microsoft og Google sælger den hver især som administreret tjeneste, og den samme database kører på en bærbar, så et skifte af udbyder er en flytning af data og ikke en omskrivning af applikationen.

Den anden er bredde. Én server dækker relationelle tabeller, JSON-dokumenter i `jsonb` -kolonner, fuldtekstsøgning, geodata gennem udvidelsen PostGIS og planlagte jobs gennem `pg_cron` . Hver af dem er et selvstændigt produkt, som et team slipper for at drive, sikre, tage backup af og ansætte folk til. For de fleste produkter afgøres PostgreSQL kontra MySQL netop her: til almindelig læsning og skrivning klarer begge sig fint, og forskellen viser sig, når arbejdet kræver rigere typer, strammere constraints eller en udvidelse.

Den tredje er levetid. Projektet udgiver én hovedversion om året og understøtter hver af dem i fem år, så en opgradering kan planlægges i stedet for at blive tvunget igennem af en nødsituation. PostgreSQL 18, udgivet i september 2025,

tilføjede asynkron I/O og en indbygget funktion til tidsordnede UUID'er, og SQL skrevet til versioner, der er et årti ældre, kører som regel uændret på den.

Kompromiserne er reelle. En proces pr. forbindelse gør forbindelser dyre, så en travl applikation har brug for en pooler. Den vokser lettere på én større maskine end på tværs af flere: at sprede skrivninger over mange servere kræver partitionering, replikering eller en udvidelse, og hver af dem lægger driftsmæssig vægt til. Og standardindstillingerne er afstemt efter en lille maskine, så en seriøs installation skal konfigureres med vilje.

Sådan ser det ud i praksis

Mange af projekterne på dette site kører på PostgreSQL, og tre af dem viser tre forskellige måder at læne sig op ad den på.

På én platform blev databasen hele datalaget. Hver læsning og skrivning går gennem en af 1.275 PostgreSQL-funktioner fordelt på 34 skemaer, og Go-API'et har ingen direkte adgang til nogen tabel. Den rolle, API'et logger ind som, `mariner`, har EXECUTE på funktionerne og USAGE på skemaerne og intet andet, cirka 4.000 eksplicitte grants. Funktionerne kører som SECURITY DEFINER under en separat ejerrolle uden login og med en fastlåst `search_path`. Adgangsgrænsen håndhæves af Postgres selv; prisen er, at et nyt felt betyder en migration og en funktionsændring.

På en applikation med tenants, lokationer og daglige elmålinger var arbejdet tuning frem for design, og målet var 10 gange bedre ydeevne uden at ændre den eksisterende arkitektur. GiST-indeks til geodata, sammensatte indeks på tenant og datointerval, range-partitionering af eltabellen pr. dag og måned og forespørgsler omskrevet efter en gennemlæsning af deres planer med `EXPLAIN ANALYZE` forbedrede svartiderne 10 gange og sænkede ressourceforbruget med 40 %.

Og på en platform med langsomt baggrundsarbejde ligger jobkøen i den samme database. River holder sin kø i PostgreSQL, så et job og de rækker, det rører, committer eller ruller tilbage sammen. Bag den ligger 15 worker-moduler, 8 planlagte opgaver og 20 `pg_cron`-jobs, der beskærer partitioner, roterer salts og opfrisker aggregater. Projektsiden siger ligeud, hvor grænsen går: en kø i Postgres holder op med at være det rigtige svar ved en vis mængde, og én bevægelig del mindre var mere værd end luft, ingen alligevel ville bruge.

Hvor det går galt

PostgreSQL mister sjældent data. Den overrasker til gengæld folk, og overraskelserne kommer i nogle få velkendte former.

Tabellen vokser, mens antallet af rækker står stille. En `UPDATE` eller `DELETE` efterlader den gamle rækkeversion, indtil `VACUUM` fjerner den, fordi en anden transaktion stadig kan være ved at læse den. En transaktion, der står åben i timevis, holder den oprydning tilbage for hele databasen, og autovacuums standardindstillinger er beregnet til små tabeller. Luk inaktive transaktioner, tilpas autovacuum pr. tabel for de store, og partitionér data, der udløber, så gamle måneder droppes som hele partitioner i stedet for at blive slettet række for række. Vacuum har også en anden opgave: transaktions-id'er er på 32 bit, hver tabel skal vacuumes mindst én gang for hver to milliarder transaktioner, og en server, der kommer for langt bagud, nægter nye skrivninger for at beskytte data.

Hukommelsen slipper op ved den første trafikspids. Hver forbindelse er en serverproces med sin egen hukommelse, og standardgrænsen er 100 forbindelser. Tyve applikationsinstanser med en pulje på halvtreds hver beder om tusind processer. Sæt en pooler som PgBouncer foran, og dimensionér hver applikations pulje efter arbejdet, hvilket som regel er en håndfuld forbindelser pr. instans.

En forespørgsel, der tog millisekunder, begynder at tage sekunder. Planneren valgte en anden plan, efterhånden som data voksede, ofte fordi dens statistik blev forældet, eller fordi indeksene var bygget til sidste års forespørgsler. Læs planen, før du ændrer noget, opdatér statistikken med `ANALYZE`, og indeksér det prædikat, forespørgslen faktisk filtrerer på:

```
EXPLAIN (ANALYZE, BUFFERS)
SELECT id, total FROM orders
WHERE customer_id = 42 AND placed_at >= '2026-01-01';
```

Inserts bliver langsommere, efterhånden som tabellen vokser. Tilfældige UUID'er som primærnøgler lander spredt over hele B-tree-indekset, så hver insert rammer en anden side, og indekset fragmenterer. Tidsordnede id'er undgår det: én platform gjorde UUID v7 til standardnøgle med `uuidv7()` fra PostgreSQL 18, og nye rækker sorterer ind i indekset i stedet for at blive drysset ud over det.

Læs mere

- What Is PostgreSQL? — projektets egen definition med de egenskaber og udvidelsespunkter, den nævner.
- A Brief History of PostgreSQL — Berkeley, Postgres95 og navneskiftet, fortalt af projektet selv.
- Concurrency Control: Introduction — hvordan øjebliksbilleder lader læsning og skrivning foregå uden at blokere hinanden.
- Routine Vacuuming — døde rækkeversioner, autovacuum og wraparound af transaktions-id'er i detaljer.
- Using EXPLAIN — sådan læser man en forespørgselsplan, med gennemgåede eksempler.
- Versioning Policy — den årlige hovedversion og de fem års support bag hver enkelt.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring et repræsentativt datasæt, de spørgsmål det skal besvare, og den aktuelle flaskehals. Vi kan aftale målinger af kvalitet, gennemløb og gendannelse.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er PostgreSQL?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

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

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/postgresql/