Engineer

Database

en ordnet samling data, som programmer kan gemme og søge i

Hvad det er

En database er en mængde data, der holdes i en fast struktur med regler for, hvad der må komme ind, så software kan finde, ændre og kontrollere indholdet; ordet, ofte forkortet DB, bruges også løst om det program, der står for opbevaringen, og som præcist hedder et databasehåndteringssystem eller DBMS. NISTs ordliste har to definitioner, og den bredeste taler om et lager af information eller data, der kan være et traditionelt relationelt databasesystem, men ikke behøver at være det.

Ordet har tre opgaver, og en samtale kører i ring, når to personer mener hver sin. Det betegner dataene: de kunder, ordrer og sensoraflæsninger, en organisation gemmer. Det betegner motoren, der passer på dem: PostgreSQL, SQL Server, SQLite. Og inde i en server betegner det én beholder blandt flere. PostgreSQLs tutorial stiller niveauerne op: rækker med navngivne, typede kolonner udgør tabeller, tabeller samles i databaser, og de databaser, som én serverinstans forvalter sammen, udgør en databaseklynge. SQL Server skelner på samme måde mellem en instans og de databaser, den rummer. »Databasen er langsom« kan betyde én tabel, én beholder eller en hel maskine.

De fleste motorer i daglig brug er relationelle: de gemmer data som relationer, den matematiske betegnelse for tabeller, og tager imod spørgsmål på SQL. PostgreSQLs tutorial nævner andre ordninger. Filerne og mapperne i et Unix-lignende system danner en hierarkisk database, og objektorienterede databaser kom til senere. Redis kalder sig selv en data structure server: i stedet for tabeller tilbyder den strenge, hashes, lister, sæt og andre indbyggede typer, der bruges til caching, køhåndtering og behandling af hændelser.

Motorer adskiller sig også i, hvor de kører. En klient/server-motor er en proces, der kører hele tiden, og som mange programmer når over et netværk. En indlejret motor er et bibliotek inde i ét program, og SQLite holder en hel database i én almindelig fil; projektet siger selv, at det konkurrerer med `fopen()` og ikke med serverdatabaser. Forskellen ses i, hvordan et program forbinder sig:

```
# klient/server: forbind til en tjeneste, der allerede kører
psql "host=db.internal dbname=shop user=app"

# indlejret: programmet åbner en fil, og motoren kører inde i det
sqlite3 shop.db
```

En tredje familie er bygget til analyse. BigQuery gemmer hver kolonne for sig, hvor mange traditionelle systemer holder en rækkes felter samlet på disken. Rækkelagring passer til at slå én post op; kolonnelagring passer til at summere ét felt over millioner af rækker, fordi databaseforespørgslen læser den kolonne og lader resten ligge.

Brug begrebet i en beslutning

Brug »Hvad er en database?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. En struktur, der passer til, hvordan forretningen faktisk fungerer, så de svar, I får brug for senere, stadig kan hentes.

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. Har designet og drevet 6 ETL/ELT-pipelines — 6 ETL-pipelines på tre databasemotorer

Har designet og drevet 6 ETL/ELT-pipelines Kontakt os →

Hvorfor det betyder noget

Filer kan også rumme data. Det, en database lægger til, er løfter om de data i det øjeblik, mange programmer bruger dem på én gang.

Det første løfte er et sammenhængende billede under samtidig brug. PostgreSQL giver hver sætning et øjebliksbillede af dataene, som de stod på et bestemt tidspunkt, et design kaldet multiversion concurrency control, eller MVCC, så læsning aldrig blokerer skrivning, og skrivning aldrig blokerer læsning. SQLite tillader et vilkårligt antal læsere ad gangen og én skriver i hvert øjeblik, mens de andre skrivere venter på deres tur. De to passer til forskellige antal skrivere, og det antal afgør meget af valget.

Det andet løfte er, at data bevarer sin form. Et skema angiver, hvilke kolonner en tabel har, og hvilken type hver rummer; nøgler og constraints angiver, hvilke rækker der må findes. Når databasen håndhæver reglerne, adlyder hvert program, der skriver til den, dem. Når reglerne kun bor i applikationskoden, efterlader det første program, der springer dem over, rækker, som hver senere forespørgsel må gardere sig imod.

Det tredje er et fælles sprog med dialekter. SQL er standarden ISO/IEC 9075, senest revideret som SQL:2023, men PostgreSQLs bilag om overensstemmelse bemærker, at intet nuværende system hævder fuld overensstemmelse med standardens Core. Et skema og dets forespørgsler hører til deres motor, og at flytte dem til en anden er en migrering med sin egen test.

SQLites egen dokumentation gør valget af motor til en kort tjekliste: data, der ligger over et netværk fra den kode, der sender SQL'en, mange processer, der skriver i samme øjeblik, eller indhold på vej mod terabyte-området peger hver for sig mod klient/server, og ellers er SQLite det enklere svar. Rapportering i stor skala rejser endnu et spørgsmål: om tunge læsninger skal køres på en særskilt analysekopi, der fyldes via ETL, så en lang rapport ikke konkurrerer med de forespørgsler, kunderne venter på.

Sådan ser det ud i praksis

I den store ende blev der i en rolle som databaseingeniør i en hurtigt voksende teknologivirksomhed oprettet, administreret og optimeret over 100 databaser i PostgreSQL, Microsoft SQL Server og Google BigQuery, primært med GIS-data og tidsseriedata som sensoraflæsninger, logs og realtidsmetrikker. Arbejdet gav normaliserede og denormaliserede skemaer til PostgreSQL og SQL Server med spatial indeksering (PostGIS på PostgreSQL-siden) og tidsseriepartitionering, mens BigQuery bar store mængder tidsseriedata i tidspartitionerede, clustrede tabeller. Med indeksering, materialiserede views og caching på plads faldt svartiderne for forespørgsler på GIS- og tidsseriedata med 40–60 %, og automatiseret overvågning reducerede nedetiden med 50 %.

På et dokumenteret udviklingsstadie lagrede et mindre system fakta bag et CV, et referenceark, en portefølje og en ansøgning i én SQLite-database på tre sprog. Én kommando byggede databasen op med alle treogtyve tabeller: den begyndte med en tom fil og kørte nitten ordnede trin, som hvert var sikkert at gentage. Enogfyrre Python-moduler, i alt 10.580 linjer, gengav seks outputformater fra den. En profil valgte, hvilke resultater der optrådte og i hvilken rækkefølge, så et

dokument rettet mod én slags læser med sidens ord var »en databaseforespørgsel og ikke en kopi«. Prisen var, at systemet derefter var den eneste måde at frembringe et dokument på. Disse tal beskriver det dokumenterede udviklingsstadie, ikke det nuværende kodelager eller de sprog, det understøtter.

Mellem de to størrelser ligger det spørgsmål, enhver travl database møder, og en note om at optimere før sharding svarer med en rækkefølge: kig, stram op, tag læsningerne væk fra den primære, partitionér tabellerne omkring det varme punkt, og shard først, når én maskine er fuld af godt arbejde. At kigge betyder at spørge PostgreSQL selv: `pg_stat_statements` for de sætninger, der koster mest i alt, og derefter `EXPLAIN (ANALYZE, BUFFERS)` for, hvad den dyreste af dem gør.

Hvor det går galt

Det meste databasebesvær begynder i en beslutning, der blev truffet tidligt, da dataene var små, og beslutningen så gratis ud.

To kopier af ét faktum glider fra hinanden. CV-generatoren ovenfor findes, fordi fakta, der holdes som fire dokumenter, skal rettes fire gange, indtil to af dem er uenige, og intet siger, hvilket der gælder. Det samme sker med en tabel, der eksporteres til et regneark og redigeres der, eller med en anden tjeneste, der fører sin egen kundeliste. Giv hvert faktum ét lager, der ejer det, og lad enhver anden kopi sige, hvor den kom fra og hvornår. Notens cache følger reglen strengt: den må fortælle en køber, at en vare er fuldt reserveret, men hvert ja kommer fra databasen.

Flere maskiner kommer, før nogen har læst planerne. At sprede data over flere servere spreder de langsomme sætninger over mere hardware uden at gøre nogen af dem billigere, og det lægger en varig regning på alt andet: som noten opregner det, kører hver migrering, backup, gendannelse og rapport én gang pr. shard, og en join på tværs af shards bliver applikationskode. Læs `pg_stat_statements` og planerne først, og gem det, de viser, som udgangspunkt.

En læsning rammer en kopi, der er bagud. En streaming-replika tager læsninger væk fra den primære og ligger en smule efter den. En bruger, der lige har skrevet noget og læser det tilbage fra replikaen, kan se den gamle værdi. Læs en brugers egne nye skrivninger fra den primære.

Rapporter og kunder deler én motor. En rapport, der scanner to års historik, beslaglægger hukommelse og disk, som butikkens egne forespørgsler har brug for, og det mærker de, der handler. Flyt tung analyse til en warehouse-kopi, og lad den daglige database tage sig af de mange små læsninger og skrivninger.

Læs mere

- NIST glossary: database — to korte definitioner fra NIST-publikationer, hvoraf den ene også rummer databaser, der ikke er relationelle.
- PostgreSQL tutorial: Concepts — tabeller, rækker, kolonner, databaser og databaseklyngen på én side.
- PostgreSQL: Introduction to MVCC — hvordan øjebliksbilleder lader læsere og skrivere arbejde uden at blokere hinanden.
- PostgreSQL: SQL Conformance — delene og versionerne af ISO/IEC 9075, og hvad Core-overensstemmelse betyder.
- Appropriate Uses For SQLite — hvor en indlejret motor passer, hvor en klient/server-motor passer bedre, og en tjekliste til valget.
- Overview of BigQuery storage — række- og kolonneorienteret lagring, og hvorfor analyse foretrækker den sidste.

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 en database?« 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/database/