Engineer

Databas

lagrade data, ordnade så att program kan ställa frågor

Vad det är

En databas är en mängd data som hålls i en fast struktur, med regler för vad som får komma in, så att program kan hitta, ändra och kontrollera innehållet. Samma ord, ofta förkortat DB, används också löst om programmet som sköter förvaringen. Det exakta namnet på det programmet är databashanterare, på engelska database management system eller DBMS. Ordlistan från NIST återger två definitioner, och den bredare beskriver ett förråd av information eller data som kan vara, men inte behöver vara, ett traditionellt relationsdatabassystem.

Ordet gör tre jobb, och ett samtal går i cirklar när två personer menar olika saker. Det står för innehållet: kunderna, beställningarna och sensoravläsningarna som en organisation sparar. Det står för motorn som lagrar dem: PostgreSQL, SQL Server, SQLite. Och inuti en server står det för en behållare bland flera. Handledningen till PostgreSQL ställer upp nivåerna: rader med namngivna, typade kolumner bildar tabeller, tabeller grupperas i databaser, och de databaser som en serverinstans hanterar tillsammans bildar ett databaskluster. SQL Server drar samma gräns mellan en instans och de databaser den rymmer. ”Databasen är långsam” kan alltså betyda en tabell, en behållare eller en hel maskin.

De flesta motorer i dagligt bruk är relationella: de lagrar data som relationer, den matematiska termen för tabeller, och tar emot SQL-frågor. Handledningen till PostgreSQL nämner andra upplägg. Filerna och katalogerna i ett Unix-liknande system bildar en hierarkisk databas, och objektorienterade databaser kom senare. Redis beskriver sig som en datastrukturserver: i stället för tabeller erbjuder den strängar, hashar, listor, mängder och andra inbyggda typer, som används för cachning, köer och händelsehantering.

Motorer skiljer sig också i var de körs. En klient/server-motor är en långlivad process som många program når över ett nätverk. En inbäddad motor är ett bibliotek inuti ett enda program, och SQLite håller en hel databas, med tabeller, index, triggers och vyer, i en enda vanlig fil. Projektet säger själv att det konkurrerar med `fopen()` snarare än med serverdatabaser. Skillnaden syns i hur ett program ansluter:

```
# client/server: connect to a service that is already running
psql "host=db.internal dbname=shop user=app"

# embedded: the program opens a file and the engine runs inside it
sqlite3 shop.db
```

En tredje familj är byggd för analys. BigQuery lagrar varje kolumn för sig, medan många traditionella system håller en rads fält samlade på disk. Radlagring passar när du slår upp en enskild post; kolumnlagring passar när du summerar ett fält över miljontals rader, eftersom databasfrågan då läser just den kolumnen och lämnar resten i fred.

Använd begreppet i ett beslut

Använd ”Vad är en databas?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. En struktur som speglar hur verksamheten faktiskt fungerar, så att de frågor du behöver svar på senare fortfarande går att ställa.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Utformade och drev 6 ETL/ELT-pipelines — 6 ETL-pipelines över tre databasmotorer

Utformade och drev 6 ETL/ELT-pipelines Hör av er →

Varför det spelar roll

Filer rymmer också data. En databas lägger till löften om innehållet, för det läge då många program använder det på samma gång.

Det första löftet är en konsistent bild vid samtidig användning. PostgreSQL ger varje sats en ögonblicksbild av datan som den såg ut vid en viss tidpunkt, en design som kallas multiversion concurrency control, MVCC, så att läsning aldrig blockerar skrivning och skrivning aldrig blockerar läsning. SQLite tillåter hur många läsare som helst samtidigt och en skrivare åt gången, medan övriga skrivare väntar på sin tur. De två passar olika antal skrivare, och det antalet gör en stor del av valet.

Det andra löftet är att datan behåller sin form. Ett databasschema anger vilka kolumner en tabell har och vilken typ var och en håller; nycklar och villkor anger vilka rader som får finnas. När databasen själv upprätthåller reglerna följer varje program som skriver till den dem. När reglerna bara finns i applikationskoden lämnar det första program som hoppar över dem rader efter sig, och varje senare databasfråga måste sedan försvara sig mot dem.

Det tredje är ett gemensamt språk, med dialekter. SQL är standarden ISO/IEC 9075, senast reviderad som SQL:2023, men bilagan om standardefterlevnad i dokumentationen för PostgreSQL noterar att inget nuvarande system gör anspråk på full överensstämmelse med standardens Core. Ett schema och dess databasfrågor hör till sin motor, och att flytta dem till en annan är ett eget projekt med egen testning.

Dokumentationen för SQLite gör valet av motor till en kort checklista. Data som ligger på andra sidan ett nätverk från koden som skickar SQL, många processer som skriver i samma ögonblick eller innehåll på väg mot terabyteklassen pekar var och en mot klient/server; annars är SQLite det enklare svaret. Rapportering i stor skala lägger till en fråga till: om tunga läsningar hör hemma på en separat analyskopia, fylld via ETL, så att en lång rapport aldrig konkurrerar med de databasfrågor som kunderna väntar på.

Så ser det ut i praktiken

Databaserna här sträcker sig från över 100 i en och samma roll till en enda SQLite-fil.

I den stora änden skapades, förvaltades och optimerades över 100 databaser i PostgreSQL, Microsoft SQL Server och Google BigQuery inom en roll som databasingenjör på ett snabbväxande teknikföretag. De flesta av dem lagrade GIS-data och tidsseriedata som sensoravläsningar, loggar och mätvärden i realtid. Arbetet gav normaliserade och denormaliserade scheman för PostgreSQL och SQL Server, med spatial indexering (PostGIS på PostgreSQL-sidan) och tidsseriepartitionering, medan BigQuery bar stora mängder tidsseriedata i tidspartitionerade, klustrade tabeller. Med indexering, materialiserade vyer och cachning på plats minskade svarstiderna för databasfrågor mot GIS- och tidsseriedata med 40–60 %, och automatiserad övervakning minskade driftstoppen med 50 %.

I ett dokumenterat utvecklingsskede lagrade ett mindre system fakta bakom ett CV, ett referensblad, en portfölj och ett personligt brev i en enda SQLite-fil på tre språk. Databasens tjugotre tabeller fylldes av en datapipeline i nitton ordnade steg, och ett enda kommando byggde hela databasen från grunden; varje steg var idempotent, alltså säkert att köra igen. Fyrtioen Python-moduler, 10 580 rader sammanlagt, renderade sex utdataformat ur den. En profil valde vilka resultat som visades och i vilken ordning, så ett dokument riktat till en viss sorts läsare var, med sidans egna ord, ”en databasfråga snarare än en kopia”. Priset var att systemet därefter var det enda sättet att ta fram ett dokument. Dessa tal beskriver det dokumenterade utvecklingsskedet, inte det nuvarande kodförrådet eller dess språkstöd.

Mellan de storlekarna står frågan som varje hårt belastad databas till slut möter, och en anteckning om att optimera innan du shardar besvarar den med en arbetsordning: titta, skärp, flytta läsningarna från primärnoden, partitionera tabellerna runt den heta punkten och sharda först när en maskin är full av bra arbete. Att titta betyder att fråga PostgreSQL själv: `pg_stat_statements` för de satser som kostar mest totalt, sedan `EXPLAIN (ANALYZE, BUFFERS)` för vad den dyraste av dem gör.

Var det går fel

De flesta problem med en databas börjar i ett beslut som fattades tidigt, när datan var liten och beslutet såg gratis ut.

Två kopior av samma uppgift glider isär. CV-generatorn ovan finns för att fakta som hålls i fyra dokument rättas fyra gånger, tills två av dem säger emot varandra och ingenting visar vilket som gäller. Samma sak händer en tabell som exporteras till ett kalkylark och redigeras där, eller en andra tjänst med en egen kundlista. Ge varje uppgift en enda plats som äger den, och låt varje annan kopia säga var den kommer ifrån och när den togs. Anteckningens cache följer den regeln strikt: den får tala om för en köpare att en vara är fullt reserverad, men varje ja kommer fortfarande från databasen.

Fler maskiner kommer innan någon har läst frågeplanerna. Att dela upp data över flera servrar sprider de långsamma satserna över mer hårdvara utan att göra någon av dem billigare, och det lägger en bestående kostnad på allt annat. Som anteckningen räknar upp körs varje databasmigrering, säkerhetskopiering,

återställning och rapport en gång per shard, och en join över flera shards blir applikationskod. Läs `pg_stat_statements` och frågeplanerna först, och behåll det de visar som utgångsläge.

En läsning hamnar på en kopia som ligger efter. En strömmande replik tar läsningar från primärnoden och ligger hela tiden en liten bit efter den. En användare som just har skrivit något och sedan läser tillbaka det från repliken kan få se det gamla värdet. Läs en användares egna nyliga skrivningar från primärnoden.

Rapporter och kunder delar en motor. En rapport som skannar två års historik håller minne och disk som butikens egna databasfrågor behöver, och de som handlar märker det. Flytta tung analys till en kopia i ett analysdatalager, och låt den operativa databasen sköta de många små läsningar och skrivningar som den finns till för.

Läs mer

- NIST glossary: database — två korta definitioner ur publikationer från NIST, varav den ena ger plats för databaser som inte är relationella.
- PostgreSQL tutorial: Concepts — tabeller, rader, kolumner, databaser och databasklustret, på en enda sida.
- PostgreSQL: Introduction to MVCC — hur ögonblicksbilder låter läsare och skrivare arbeta utan att blockera varandra.
- PostgreSQL: SQL Conformance — delarna och versionerna av ISO/IEC 9075, och vad överensstämmelse med Core innebär.
- Appropriate Uses For SQLite — var en inbäddad motor passar, var en klient/server-motor passar bättre, och en checklista för valet.
- Overview of BigQuery storage — rad- och kolumnorienterad lagring, och varför analys gynnar den senare.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med ett representativt dataset, frågorna det ska besvara och dagens flaskhals. Vi kan avtala mått på kvalitet, genomströmning och återställning.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är en databas?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

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

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/database/