Engineer

SQL

sproget, relationelle databaser svarer på

Hvad det er

SQL er et deklarativt sprog til at definere, ændre og læse data, der ligger i relationelle tabeller: Man beskriver det resultat, man vil have, og databasen afgør selv, hvordan det skal frembringes. Forkortelsen udfoldes som regel som Structured Query Language og udtales enten bogstav for bogstav eller som »sequel«, mens den internationale standard, ISO/IEC 9075, blot kalder det Database Language SQL.

Det deklarative er hele pointen. Sætningen nedenfor nævner en tabel, en betingelse, en gruppering og en rækkefølge – men siger intet om indeks, løkker eller hvilke rækker der skal læses først:

```
SELECT land, count(*) AS tilmeldinger
FROM konti
WHERE oprettet >= now() - interval '30 days'
GROUP BY land
ORDER BY tilmeldinger DESC;
```

I PostgreSQL går teksten gennem fire trin. En parser kontrollerer syntaksen og bygger et træ, et omskrivningstrin folder de views ud, som sætningen nævner, en planlægger sammenligner de mulige måder at udføre den på – for eksempel en sekventiel scanning over for et indeks – og til sidst henter en executor rækkerne. Den samme sætning kan tage et millisekund eller et minut, alt efter hvad planlæggeren finder, og derfor hører det at læse en plan med til at skrive SQL.

Sproget har dele, som man ofte nævner hver for sig. Datadefinition opretter og ændrer tabeller, skemaer og indeks. Datamanipulation indsætter, opdaterer, sletter og fletter rækker. En SQL-forespørgsel læser dem med SELECT.

Rettigheder og transaktioner afgør, hvem der må hvad, og hvilke ændringer der lander samlet. Det hele sendes som tekst over én forbindelse, så SQL er fællesgrunden mellem en applikation, et rapportværktøj og et menneske ved en terminal.

Standarden er revideret mange gange, senest som SQL:2023, og hver motor implementerer sin egen delmængde af den. PostgreSQL's bilag om standardoverensstemmelse oplyser, at systemet opfylder mindst 170 af de 177 obligatoriske Core-egenskaber, og at intet databasesystem på skrivetidspunktet gjorde krav på fuld Core-overensstemmelse. Oven i det lægger hver motor sin egen dialekt: Microsoft SQL Server taler Transact-SQL, BigQuery taler GoogleSQL, og PostgreSQL tilføjer PL/pgSQL til procedurekode, der kører inde i databasen.

Brug begrebet i en beslutning

Brug »Hvad er SQL?« 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

Det valg, SQL stiller et team over for, er, hvor arbejdet med data skal foregå: inde i databasen, i det sprog den er bygget til at udføre, eller i applikationskode, der henter rækkerne og behandler dem selv.

En mængdebaseret sætning kører lige ved siden af data. Filtrerer man en million rækker ned til fyrre inde i databasen, går der fyrre rækker over netværket; filtrerer man i applikationen, går der en million, og de fleste smides væk. Joins, aggregeringer og fjernelse af dubletter følger samme regnestykke. At flytte den slags arbejde ind i SQL er ofte den billigste hastighedsforbedring, et langsomt system har, og det kræver ingen ny infrastruktur.

Modvægten er ejerskab. Logik skrevet i SQL bor i migreringer og funktioner, der kræver samme review, test og versionsstyring som al anden kode, og færre udviklere læser en lang PL/pgSQL-funktion lige så flydende som Go eller Python.

En brugbar fordeling sender det, der skal se mange rækker, til databasen og beholder de regler, der ændrer sig med forretningen, i applikationen, hvor de er lettere at teste isoleret.

Portabilitet er den anden afvejning. Kernen af sproget – SELECT med joins, GROUP BY og ORDER BY – flytter mellem motorer med få rettelser. Kanterne flytter dårligt: Datoregning, upserts, paginering, JSON-funktioner og procedurekode er forskellige fra motor til motor, og en migrering fra én database til en anden bruger det meste af kræfterne netop dér. Vælger man en motor, vælger man også en dialekt, og jo mere logik der bor i dialekten, jo dyrere bliver et senere skifte.

Rekruttering er den tredje. SQL er en af de få færdigheder, som analytikere, backendudviklere, dataingeniører og databaseadministratorer har til fælles, så et datalag skrevet i almindelig SQL kan læses og reviewes af dem alle. En ORM skjuler SQL'en, indtil en side bliver langsom, og så skal nogen alligevel læse, hvad den genererede.

Sådan ser det ud i praksis

På én platform samler PostgreSQL-funktioner hvert API-svar som et færdigt JSON-dokument, og Go-handleren sender det videre som `json.RawMessage` uden at afkode eller genkode det. Med JSON-passthrough-arkitekturen blev formningen af svaret en SQL-opgave, løst dér, hvor data allerede ligger. En lille QueryJSON-hjælper gjorde mønstret til det letteste at følge, og de DTO'er, mappere og svarstrukturer, som et lagdelt API normalt samler op, forsvandt. Handlerkoden blev meget kortere, og en ændring i det, en funktion returnerer, når frem til klienten, uden at nogen retter i en handler.

I en datapipeline ser det samme sprog anderledes ud. På tværs af seks ETL/ELT-pipelines med BigQuery, MSSQL og PostgreSQL blev rå filer indlæst i staging-skemaer med bcp, psql og SSIS, mens rensning, normalisering og berigelse blev skrevet som PL/pgSQL- og T-SQL-procedurer. Tre motorer med hver sin dialekt i ét job: portabilitetsspørgsmålet ovenfor, set indefra. Siden gør resultatet op sådan: hvor jobbet før krævede mindst tre timers manuelt arbejde, tog hele kørslen bagefter under 20 minutter, data blev opdateret dagligt i stedet for ugentligt, og de Python-API'er, der læser dem, blev 30 % hurtigere.

Hvor det går galt

Sætninger limet sammen af strenge. En kunde ved navn O'Brien får en søgeside til at fejle, eller et udspekuleret input læser andres data. Årsagen er brugerinput, der er klistret ind i SQL-teksten, hvor databasen ikke kan skelne data fra kommando. Send i stedet værdierne som parametre, adskilt fra sætningen, som alle udbredte drivere tillader. I PostgreSQL's eget klientbibliotek tager et parametriseret kald desuden kun én kommando pr. streng, hvilket giver en ekstra forsvarslinje mod SQL-injection.

Rækkefølge, der kun var held. En liste, der har været sorteret i månedsvis, kommer pludselig blandet efter en dataindlæsning eller en ny plan. Uden ORDER BY returnerer PostgreSQL rækkerne i den rækkefølge, der er hurtigst at frembringe, og en LIMIT uden ORDER BY kan endda give en anden delmængde hver gang. Ethvert resultat, hvis rækkefølge betyder noget, skal sige det i forespørgslen.

NULL behandlet som en værdi. Et filter skrevet som `WHERE rabat = NULL` returnerer ingen rækker, og `NOT IN` mod en delforespørgsel, der indeholder en NULL, gør det samme. Enhver sammenligning med NULL giver ukendt i stedet for sand eller falsk, og en WHERE-klausul beholder kun de rækker, hvor betingelsen er sand. Skriv `IS NULL`, og foretræk `NOT EXISTS` frem for `NOT IN`, når den indre kolonne kan indeholde NULL.

Dialekt, der siver ind i testene. En testsuite består mod SQLite, og samme kode fejler mod PostgreSQL i produktion – eller en flytning til SQL Server snubler over hver eneste `LIMIT`, som den motor staver `TOP` eller `OFFSET ... FETCH`. Hver motor accepterer ting, som de andre afviser. Test mod den motor, I faktisk kører, og hold motorspecifik syntaks samlet få steder.

Læs mere

- Part II. The SQL Language — PostgreSQL's samlede vejledning og opslagsværk for sproget, fra syntaks til ydelsestips.
- Appendix D. SQL Conformance — standardens dele, og hvilke af dens egenskaber PostgreSQL implementerer.
- Transact-SQL Reference — SQL Server-dialekten, dokumenteret af Microsoft.
- Introduction to SQL in BigQuery — GoogleSQL og den ældre dialekt, den afløste.

- Query Language Understood by SQLite — syntaksen i en indlejret motor, sætning for sætning.

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 SQL?« 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/sql/