Engineer

SQL

språket som relationsdatabaser svarar på

Vad det är

SQL är ett deklarativt språk för att definiera, ändra och läsa data som ligger i relationstabeller: du skriver ned resultatet du vill ha, och databasen avgör hur det ska tas fram. Bokstäverna brukar utläsas Structured Query Language och uttalas antingen bokstav för bokstav eller som ”sequel”, medan den internationella standarden som definierar språket, ISO/IEC 9075, helt enkelt kallar det Database Language SQL. En enskild sats som läser data kallas ofta en SQL-fråga.

Deklarativ är ordet som gör jobbet. Satsen nedan namnger en tabell, ett villkor, en gruppering och en ordning. Den säger ingenting om index, loopar eller vilka rader som ska läsas först:

```
SELECT country, count(*) AS signups
FROM accounts
WHERE created_at >= now() - interval '30 days'
GROUP BY country
ORDER BY signups DESC;
```

I PostgreSQL passerar den texten fyra steg. En parser kontrollerar syntaxen och bygger ett träd, ett omskrivningssteg expanderar de vyer som satsen nämner, en planerare jämför de möjliga sätten att köra den, t.ex. en sekventiell genomläsning mot ett index, och en exekverare hämtar till sist raderna. Samma mening kan ta en millisekund eller en minut beroende på vad planeraren hittar, och därför hör det till att skriva SQL att kunna läsa en plan.

Språket har delar som brukar få egna namn. Datadefinition skapar och ändrar tabeller, scheman och index. Datamanipulation infogar, uppdaterar, raderar och slår ihop rader. Frågor läser dem med SELECT. Behörigheter och transaktioner avgör vem som får göra vad och vilka ändringar som landar tillsammans. Allt

detta färdas som text över en enda anslutning, och därför är SQL den gemensamma marken mellan en applikation, ett rapportverktyg och en människa vid en terminal.

Standarden har reviderats många gånger, senast som SQL:2023, och varje databasmotor implementerar sin egen delmängd av den. Bilagan om överensstämmelse i PostgreSQL anger att motorn uppfyller minst 170 av de 177 obligatoriska Core-funktionerna, och att inget databassystem när texten skrevs påstod sig uppfylla hela Core. Varje motor lägger dessutom till en egen dialekt: Microsoft SQL Server talar Transact-SQL, BigQuery talar GoogleSQL, och PostgreSQL lägger till PL/pgSQL för procedurkod som körs inuti databasen.

Använd begreppet i ett beslut

Använd ”Vad är SQL?” 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

Det beslut som SQL ställer ett team inför är var arbetet med data sker: inuti databasen, i det språk den byggdes för att köra, eller i applikationskod som hämtar rader och bearbetar dem själv.

En mängdbaserad sats körs bredvid datan. Att filtrera en miljon rader ned till fyrtio inuti databasen skickar fyrtio rader över nätverket; att filtrera i applikationen skickar en miljon och kastar de flesta. Joins, aggregat och borttagning av dubbletter följer samma aritmetik. Att flytta den sortens arbete in i SQL är ofta den billigaste uppsnabbning som ett långsamt system har tillgång till, eftersom den körs på den databas som redan finns.

Ägarskapet är motvikten. Logik som skrivs i SQL bor i migreringar och funktioner, som behöver samma granskning, samma tester och samma versionshantering som all annan kod, och färre utvecklare läser en lång PL/pgSQL-funktion lika obehindrat som de läser Go eller Python. En fungerande uppdelning skickar allt som måste se många rader till databasen och behåller regler som ändras med verksamheten i applikationen, där de är lättare att testa var för sig.

Portabiliteten är den andra avvägningen. Kärnan i språket, SELECT med joins, GROUP BY och ORDER BY, flyttar mellan motorer med få ändringar. Kanterna flyttar dåligt: datumaritmetik, upserts, paginering, JSON-funktioner och procedurkod skiljer sig alla åt, och en migrering från en databas till en annan lägger det mesta av sin möda där. Att välja motor är också att välja dialekt, och ju mer logik som bor i den dialekten, desto mer kostar en senare flytt.

Rekryteringen är den tredje. SQL är en av få färdigheter som analytiker, backendutvecklare, dataingenjörer och databasadministratörer delar, så ett dataåtkomstlager skrivet i ren SQL kan läsas och granskas av vem som helst av dem. En ORM döljer SQL tills en sida blir långsam, och då måste någon ändå läsa vad den genererade.

Så ser det ut i praktiken

På en plattform sätter PostgreSQL-funktioner ihop varje svar från API:et som ett komplett JSON-dokument, och Go-handlern skickar det vidare som `json.RawMessage` utan att avkoda eller koda om det. Arkitekturen som skickar JSON rakt igenom gör formningen av svaret till en fråga för SQL, utförd där datan redan finns. En liten hjälpfunktion, QueryJSON, gjorde mönstret till det lättaste att följa, och de DTO:er, mappningsfunktioner och svarsstructar som ett lagerindelat API brukar samla på sig försvann. Handlerkoden blev mycket kortare, och en ändring av vad en funktion returnerar når klienten utan att någon redigerar en handler.

Samma språk ser annorlunda ut i en datapipeline. I sex ETL/ELT-pipelines som spände över BigQuery, MSSQL och PostgreSQL lästes råfiler in i scheman för mellanlagring med bcp, psql och SSIS, och rensningen, normaliseringen och berikningen skrevs som procedurer i PL/pgSQL och T-SQL. Tre motorer, var och en med sin egen dialekt, i ett och samma jobb: frågan om portabilitet ovan, sedd inifrån. Sidan anger resultatet som minst tre timmars manuellt arbete som krympte till under 20 minuter för hela flödet, data som uppdaterades dagligen i stället för veckovis och Python-API:er som läste dessa data och blev 30 % snabbare.

Var det går fel

Satser som limmas ihop av strängar. En kund som heter O’Brien får en söksida att gå sönder, eller en konstruerad inmatning läser någon annans data. Orsaken är att användarens indata klistras in i SQL-texten, där databasen inte kan skilja

data från kommando. Skicka i stället värden som parametrar, separat från satsen, vilket varje etablerad drivrutin tillåter. I klientbiblioteket som PostgreSQL själv levererar accepterar ett parametriserat anrop dessutom bara ett kommando per sträng, vilket ger en andra försvarslinje mot injektion.

Ordning som bara var tur. En lista som har kommit tillbaka sorterad i månader dyker upp omkastad efter en datainläsning eller en ny plan. Utan ORDER BY returnerar PostgreSQL rader i den ordning som går snabbast att ta fram, och en LIMIT utan ORDER BY kan till och med returnera en annan delmängd varje gång. Varje resultat vars ordning spelar roll säger det i frågan.

NULL som behandlas som ett värde. Ett filter skrivet `WHERE discount = NULL` returnerar ingenting, och `NOT IN` mot en delfråga som innehåller en NULL returnerar heller ingenting. Varje jämförelse med NULL ger okänt snarare än sant eller falskt, och en WHERE-sats behåller bara de rader där villkoret är sant. Skriv `IS NULL`, och föredra `NOT EXISTS` framför `NOT IN` när den inre kolumnen kan innehålla en NULL.

Dialekt som läcker in i testerna. En testsvit går igenom mot SQLite och samma kod fallerar mot PostgreSQL i produktion, eller så snubblar en flytt till SQL Server på varje `LIMIT`, som den motorn stavar `TOP` eller `OFFSET ... FETCH`. Varje motor accepterar saker som de andra avvisar. Testa mot den motor som du driftsätter, och håll motorspecifik syntax på så få ställen som möjligt.

Läs mer

- Part II. The SQL Language — den fullständiga handledningen och referensen för språket från PostgreSQL, från syntax till tips om prestanda.
- Appendix D. SQL Conformance — delarna av ISO/IEC 9075 och vilka funktioner PostgreSQL implementerar.
- Transact-SQL Reference — dialekten i SQL Server, dokumenterad av Microsoft.
- Introduction to SQL in BigQuery — GoogleSQL och den äldre dialekt som den ersatte.
- Query Language Understood by SQLite — syntaxen i en inbäddad motor, sats för sats.

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 SQL?” 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/sql/