SQL

Lär dig SQL

10. Grundläggande begrepp och smarta knep

www.databi.se

Innehåll

1. Vad är SQL?

Man skriver SQL för att ge en dator instruktioner om att hämta eller göra något med data i en relationsdatabas. Man säger oftast att man skriver SQL, men det finns olika typer av SQL-kommando eller statements och det kan vara svårt att förstå vad man faktiskt behöver lära sig.

SQL används för att skapa och hantera en relationsdatabas och tabeller, ge åtkomst till användarkonto och för att arbeta med själva datat. För att ställa frågor till databasen, t ex ”ta fram alla kunder som gjorde ett köp förra veckan” används kommandon i av den typ som kallas för Data Query Language, DQL.

Egentligen spelar det inte så stor roll vilken typ av SQL kommando man använder, men det kan vara bra att förstå att man inte behöver lära sig allt för att kunna jobba med data. DQL och lite DDL, Data Defintion Language i form av CREATE (skapa) kommer man väldigt långt på, de flesta som jobbar med data behöver aldrig hantera en databas eller användarkonton.

2. Vad är en relationsdatabas?

En relationsdatabas är en databas där data är uppdelad i tabeller, strukturerade i rader och kolumner. En tabell kan innehålla hundratals kolumner, eller bara ett fåtal, miljoner med rader eller betydligt färre.

När ett system genererar data, sparas datat i olika tabeller, men det finns oftast en gemensam nämnare som gör att man kan koppla ihop dem till en helhet. När du t ex köper en produkt, så kan informationen om köpet sparas i en tabell och dina kunduppgifter i en annan. Dina kunduppgifter kan också delas upp i olika tabeller och det finns oftast även tabeller som har information om alla produkter, deras färg, artikelnummer osv.

Alla databaser ser olika ut och man använder SQL för att skriva kod, s k queries, för att ta fram den data som man behöver. Databaserna kan finnas i molnet, på egna, lokala servrar i något som kallas för on-prem eller blandat. Vart datat lagras spelar inte så stor roll för den som jobbar med datat, så länge man får access.

3. SQL statements och syntax

Kommandona i SQL följer en viss syntax, d.v.s. en uppsättning regler för hur du ska skriva och strukturera kommandona för att de ska fungera korrekt. Man kan tänka på dem som meningar och det gäller bara att lära sig vilka ord och i vilken ordning som bildar en komplett mening, precis som vilket språk som helst. Det finns också ofta många olika sätt att säga samma sak, precis som i vanliga språk.

Exempel på SQL-kommandon är SELECT för att hämta data. Den behöver kombineras med information om vad som ska hämtas och från vart, exempelvis SELECT * FROM tabellnamn, vilket betyder HÄMTA allt(*) FRÅN tabellen-det-ska-hämtas-ifrån.

Det finns många olika former av SQL, nästan som dialekter. Dessa skiljer sig åt, men är i grunden ganska lika, lär man sig en form är det ganska enkelt att lära sig vilken syntax som gäller för de andra.

4. Pseudokod

Pseudokod är en förklaring av logiken som ska utföras, men i vanliga ord. Genom att först skriva upp stegen i det du vill ska hända, kan det bli enklare att få fram koden. Kod handlar ju mycket om logik och det är lätt att snurra till det, speciellt om koden är lång och komplicerad. Genom att använda pseudokod kan det också vara enklare att förklara för andra vad man vill uppnå.

Vårt nybörjartips är att börja en kodövning med att skriva upp det som ska kodas fram och sedan kommentera bort det genom att börja texten med /** och avsluta med */

Exempel

/** ta fram kundnummer, kundnamn, mail från kundtabell, joina in mailadresser på de som har gjort köp senaste året*/ eller
/** ta fram alla som gjort köp senaste året och deras kundnummer, kundnamn från kundtabell, joina in mailadresser*/

Genom en psudokod blir det uppenbart att det antagligen rör sig om en tabell med köp, en tabell med kunder och en med mailadresser.

5. Datatyper

MNågot man ska ha i åtanke när man börjar skriva SQL (eller andra språk som DAX eller Python) är att data kan sparas som olika datatyper. Det finns många olika typer, men det vanligaste är att man tänker på dem som

- text
- siffror
- boolean (sant eller falskt)
- decimal
- eller datum/tid

Försöker man t ex skapa en relation mellan siffrorna 111 och texten ”111” kommer det inte att fungera då den ena är just siffror och den andra text, trots att de kan se exakt likadana ut i tabellen.

När man tar fram resultat, t ex genom division, kan man få fram resultat med väldigt många decimaler. För att avrunda, kan man t ex använda kommandot ROUND(), men det finns även andra sätt och kommandon.

6. Knep för att lära sig

För att lära sig SQL finns det några smarta knep (utöver att öva, öva, öva). Vår favorit är att jobba med en liten mängd av resultatet och att kombinera SQL och Excel för att kontrollera resultatet.

1. Använd LIMIT 10 eller TOP 10 i din kod för att få fram ett litet resultat (välj antal som passar)
2. Skriv kod tills du tror att du får fram rätt resultat.
3. Exportera ditt resultat och öppna upp i Excel
4. Använd filter, pivottabell och andra knep för att ”titta” på ditt resultat.
5. Blev det fel, t ex att något summerades på fel sätt, testa att skriv logiken som Excelformel först. Även för detta kan du först exportera en sample (t ex 100 rader) från databasen. Ibland är Excel lättare, för man ser resultaten direkt.

1. Funktioner och kommando

En **funktion** är en text som genomför något, t ex SUM, som summerar de siffror den får, COUNT, som räknar ihop antal.
De flesta system använder ungefär samma text för funktioner. I SQL skriver man det som SUM(kolumnnamn), då summeras alla siffror i den kolumnen, d.v.s. det som är inom parentes.

Kommando, eller **statement**, är själva "grejen" SQL ska genomföra. De första man lär sig är SELECT, FROM, WHERE, ORDER BY, SUM, COUNT, AS och GROUP BY och JOIN.

Man kan redan från orden höra vad de används till och SQL är på så sätt ett ganska lätt språk, man skriver i princip det man vill hända i den ordningen det ska hända. Tänker man i stegvis pseudokod och skriver i den ordningen är det lätt att komma igång. Sen blir det svårare ju krångligare man tänker.

1. Funktioner och kommando

- SELECT - VÄLJ
- FROM - FRÅN (tabell)
- WHERE - VART
- ORDER BY - SORTERA
- SUM - SUMMERA
- COUNT = RÄKNA ANTAL
- AS = SOM
- GROUP BY = GRUPPERA
- JOIN = KOMBINERA/GÖRA SÄLLSKAP

8. Lättare att läsa än att skriva

Precis som vanliga språk är det lättare att förstå än att skriva eller pratat själv. En fördel är att man kan lära sig genom att undersöka andras kod, t ex på en arbetsplats. Det var så vi en gång började lära oss (innan det fanns tutorials på nätet).
Testa och förstå koden nedan, förstår du ungefär?

```
SELECT
c.customer_name AS kundnamn
,c.cutomer_id AS kundid
,e.customer_email AS email
FROM customers c
JOIN customer_emails e
ON c.customer_id = e.customer_id
```

Kika på nästa sida för facit

Kommatecken används för att separera rader

SQL

8.

Facit

```
SELECT
c.customer_name AS kundnamn
,c.cutomer_id AS kundid
,e.customer_email AS email
FROM customers c
JOIN customer_emails e
ON c.customer_id = e.customer_id
```

VÄLJ
kolumnen customer_name från tabellen customers, som jag kallar för c för att slippa skriva ut hela namnet varje gång, och kalla kolumnen för kundnamn
,kolumnen customer_id från tabellen c, (customer-tabellen), och kalla kolumnen för kundid
,kolumnen email från tabellen e (email-tabellen) och kalla kolumnen för kundid
Utgå FRÅN tabellen customers som jag i denna kodsnutt ska kalla för c, så jag slipper skriva ut ”customer” varje gång jag vill använda den tabellen.
För att lägga till email behöver den hämtaa från customer_emails tabellen, alltså joina dem.
Använd kundid från c (customers) och kundid från e (customer_emails) och jämför rad för rad för att se om det är en matchning. När det är en match, skriv mailadressen bredvd kundid.

9. JOIN data från flera tabeller

Syntax med JOIN brukar kännas lite krångligt till en början, ofta för att man gärna pratar både LEFT JOIN, LEFT OUTER JOIN, RIGHT JOIN, INNER JOIN och CROSS JOIN innan man ens förstått JOIN.

Vårt tips är att endast fokusera på LEFT JOIN till en början och först när du märker att LEFT JOIN inte räcker, då börjar du testa med övriga.

I en JOIN tar man kolumner och rader från mer än en tabell och slår ihop till en ny tabell, d.v.s. resultatet av din query. I en LEFT JOIN utgår man från den tabell som man väljer först och oavsett om det finns en matchning i den andra tabellen, så behåller man informationen.

I SQL skriver man nedåt i sitt skärmfönster för att det ska vara enklare att läsa, men man skulle också kunna skriva i en rad (men gör inte det:).

SQL

9. Exempel JOIN

Om din kod är

```
SELECT
*
FROM tabell a
LEFT JOIN tabell b
ON a.kolumn1 = b.kolumn1
```

- Då är tabell a din vänstra tabell, alltså den som du har som bas och där alla rader ska finnas kvar.
- För varje rad i tabell a, tittar man vad man har för värde i kolumn1 och slår upp i tabell b, i kolumn1.
- Det kan underlätta att tänka på dessa tabeller som något man känner till och ska mappa ihop, t ex en lista över föräldrar och barn.

* betyder alla kolumner

10. ChatGPT

Hur är det egentligen med ChatGPT eller annan generativ AI och SQL. Kan ChatGPT skriva ihop allt jag behöver? Kan man lära sig bara genom ChatGPT?

Vår erfarenhet är att ChatGPT är ett jättebra stöd, men man behöver kunna en del själv för att kunna ställa rätt frågor - alltså skriva rätt prompt.
Den är också väldigt bra på att förklara, men det kan vara svårt att förklara exakt vilket sammanhang frågan ställs i, så ibland blir förklaringarna mest förvirrande.

En stor nackdel med ChatGPT, är att det kan verka lätt att slippa lära sig. Visst klarar man sig, men en stor del av arbetet med SQL handlar om att preparera data och kontrollera att det är som det var tänkt. Dålig data leder till dåliga rapporter och även AI använder data i modellerna, så SQL kunskap är inte bara rapporter och analys “old school”.

www.databi.se