Introduktion till SQL

Structured Query Language

EVERYLOOP

www.everyloop.com

Relationsdatabaser

- En databas är en samling information som är organiserad för att man enkelt ska kunna söka och ändra enskilda delar av informationen.
- En databashanterare (DBMS) är den mjukvara som används för att söka och manipulera data i en specifik typ av databas.
- Det finns olika sorters databaser. Vi kommer fokusera på den kategori av databaser som kallas relationsdatabaser, som är väldigt vanliga.
- Relationsdatabaser lagrar data i tabellform, och ofta är det relationerna mellan olika tabeller som är det mest intressanta. Därav namnet relationsdatabas.

Structured Query Language (SQL)

- SQL är ett standardiserat programspråk för att hämta och modifiera data i relationsdatabaser (RDBMS).
- Språket utvecklades först av IBM under 70-talet.
- SQL uttalas vanligen bokstav för bokstav, alltså S-Q-L, men ibland hör man det även uttalas som engelskans ”sequel”.
- Olika databashanterare (t.ex Oracle, Postgres och MySQL) använder olika dialekter av SQL. Variationerna är dock oftast relativt små.
- I denna och kommande lektioner använder vi T-SQL som är den variant som Microsoft använder i sin SQL-server.

Queries

- För att hämta ut eller ändra information i en relationsdatabas skickar man så kallade queries (frågor) till databashanteraren (servern).
- En query kan vara allt från väldigt simpel (”ge mig all data i tabell xyz”) till väldigt komplex (t.ex korsreferera data från flera tabeller med en mängd vilkor för exakt vilka data som man vill få ut).
- I denna lektion kollar vi på några av de vanligaste uttryck man använder i SQL för att plocka ut data från en tabell.

SELECT

När vi vill hämta och visa data från en tabell använder vi ”SELECT”:

Syntax: SELECT *(lista med) kolumnnamn* FROM *tabellnamn*

SELECT id, username, password FROM users;

Id	Username	Password
1	Admin	Abc123
2	Fredrik	Secret

För att ta ut **alla** kolumner i en tabell kan man skriva:
SELECT * FROM *tabellnamn;*

TOP x *

Man vill i princip aldrig be om all data i en tabell (SELECT *) då t.ex. tabellen ”users” kan innehålla tusentals användare.

Ett sätt att begränsa antal rader är genom ”TOP x” som begränsar resultatet till x rader.

```
SELECT TOP 10 * FROM USERS;
```

WHERE

Ett annat sätt att begränsa är att bara be om rader som matchar ett givet vilkor. Detta gör man med ”WHERE”.

SELECT * FROM USERS WHERE username = ’Fredrik’;

Id	Username	Password
2	Fredrik	Secret

Logiska operationer

Select * from cities where country = ’Sweden’ **and** population < 10000;

Select * from cities where country = ’Sweden’ **or** country = ’Norway’;

Select * from cities where **not** country = ’Sweden’;

Select * from cities where country <> ’Sweden’;

In

I exemplet med ”or” ovan så tog vi ut alla rader där country är ’Sweden’ eller ’Norway’. Ett smidigare sätt är att använda ”in”.

Select * from cities where country in (’Sweden’, Norway’, ’Denmark’);

Man kan även använda ”not in” för att få alla städer från tabellen som inte ligger i de länder man anger.

Between

Man kan även använda **between** för att ange ett spann av värden:

Select * from users where id between 18 and 24;

Select * from airports where IATA between ’AAF’ and ’AAJ’;

Like

”Like” används när man vill matcha textfält mot ett specifikt mönster.
T.ex alla textfält som börjar på ’B’.

Uttrycken i tabellen kan kombineras till ett mönster som beskriver data.

Uttryck	Beskrivning
%	Vilken sträng av tecken som helst.
_	Ett tecken, vilket som helst.
[]	Ett tecken i ett set [abcd] eller spann [a-d].
[^]	Ett tecken, ej i set [^abc] eller spann [^a-c].

```
select * from cities where city like ’[a-f]%[^0-9]’;
select * from airports where IATA like ‘[acf]_[q-z]’;
select * from cities where country not like ’%land%’;
```

Order by

Man kan välja att sortera resultatet på en eller flera kolumner.

Om man t.ex vill sortera de rader man tar ut från tabellen ”people” på efternamn, och i andra hand på förnamn (i de fall flera personer har samma efternamn) kan man skriva:

```
select * from people order by lastname, firstname;
```

Efter varje kolumn vi sorterar på så kan vi lägga till ”asc” (ascending) eller ”desc” (descending) för att ange om det ska vara stigande eller fallande ordning. Anger man inget så används ”asc” som default.

Distinct

Vill man bara ha ut unika värden så kan man använda ”distinct”.

Om vi t.ex. vill lista vilka länder som finns representerade i tabellen cities utan att få en ut dubbletter på länderna i de fall de finns med på flera rader så kan vi skriva:

```
Select distinct country from cities;
```

Aliases

Aliases används för att i sin query referera till t.ex kolumner eller tabeller med ett annat namn än de har i databasen. Detta kan vara användbart t.ex för att slippa skriva så mycket, eller för att man vill att resultatet ska visas med ett annat namn.

select id, username as user, password as key from users;

id	user	key
1	Admin	Abc123
2	Fredrik	Secret

Union all

Ibland vill man slå ihop resultat från två (eller fler) queries till ett enda. Det kan man göra med ’Union all’ förutsatt att kolumnerna i de olika frågorna matchar:

```
select stad as ’city’, [invånare] as ’population’ from SvenskaOrter
union all
select city, population from UScities;
```

Om man använder bara ”union” istället för ”union all” så elimineras alla dubbletter, vilket i princip blir som att använda distinct på resultatet.

Case when

Med konstruktionen **case-when** kan man välja att visa olika värden i resultatet beroende på vilkor man angett (som utvärderas per rad).

```
Select
        City,
        case
                when population < 1500 then 'Village'
                when population < 50000 then 'Town'
                else 'City'
        end as 'Classification'
From UScities;
```

SQL är mer än bara select-satser

Man kan göra mycket mer med SQL än att bara plocka ut data från tabeller som vi gjort i exemplen ovan. Några viktiga användningsområden som vi kommer kolla på framöver är:

- Lägga till, uppdatera och ta bort rader i tabeller.
- Skapa och ta bort tabeller och så kallade vyer.
- Plocka ut data på aggregerad nivå.
- Korsreferera data från flera tabeller.
- Skriva funktioner och procedurer med flödeslogik som mer liknar ”vanlig” programmering så som vi är vana vid från t.ex c#.