

Introduktion till SQL



IT-HÖGSKOLAN

Här startar din IT-karriär.

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. Därför 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 villkor 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 [First Name], "Last Name", City  
FROM Employees;
```

▲ Result Set (15 items) ...		
First Name	Last Name	City
Nancy	Davolio	Seattle
Andrew	Fuller	Tacoma
Janet	Leverling	Kirkland



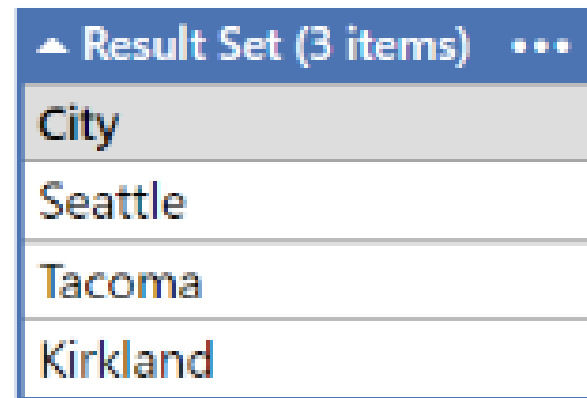
IT-HÖGSKOLAN

Här startar din IT-karriär.

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 3 City  
FROM Employees;
```



A screenshot of a SQL query result set. The title bar reads "Result Set (3 items)" with a dropdown arrow and three dots. The table has one column labeled "City" and three rows of data: "Seattle", "Tacoma", and "Kirkland".

City
Seattle
Tacoma
Kirkland

WHERE

- Ett annat sätt att begränsa är att bara be om rader som matchar ett givet villkor. Detta gör man med "WHERE".
- `SELECT *`
- `FROM Employees`
- `WHERE City = 'Seattle';`

Employee ID	Last Name	First Name	Title	Birth Date	Hire Date	Address	City	Region	Postal Code	Country	Home Phone	Extension	Photo	Notes
1	Davolio	Nancy	Sales Representative	1948-12-08 00:00:00	1991-03-29 00:00:00	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122	USA	(206) 555-9857	5467	byte[]	Educational comple
8	Callahan	Laura	Inside Sales Coordinator	1958-01-09 00:00:00	1993-01-30 00:00:00	4726 - 11th Ave. N.E.	Seattle	WA	98105	USA	(206) 555-1189	2344	byte[]	Laura re comple



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';
```



IT-HÖGSKOLAN

Här startar din IT-karriär.

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.

```
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%';
```



IT-HÖGSKOLAN

Här startar din IT-karriär.

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 dubletter på länderna i de fall de finns med på flera rader så kan vi skriva:

```
SELECT DISTINCT country FROM cities;
```



IT-HÖGSKOLAN

Här startar din IT-karriär.

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 dubletter, 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 villkor 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#.

