Laboration 2, Databashantering med MySQL

Av: Marcus Rejås <marcus@rejas.se>

> *I denna laboration skall vi jobba vidare på bildatabasen som vi började på förra gången. Vi skall ändra fält och lära oss att välja poster med **SELECT**.*
>
> *I denna laboration skriver jag inte ut resultatet på de flesta frågorna utan du kör dem själv i din miljö.*

Från förra laborationen ...

Om du gjorde den förra laborationen rätt så skall du ha en tabell med följande fält.

```
mysql> EXPLAIN bilar;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| reg       | char(10) | YES  |     | NULL    |       |
| marke     | char(50) | YES  |     | NULL    |       |
| modell    | char(50) | YES  |     | NULL    |       |
| arsmodell | int(11)  | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
4 rows in set (0.03 sec)
```

Denna tabell skall vi nu bygga vidare på, MySQL har en egenskap som är väldigt bra, nämligen att man kan uppdatera ett schema trots att det finns data i tabellen. Naturligtvis kan man inte ändra det så att det data som finns i tabellen inte passar in.

Fältet ”reg”

I fältet reg skall vi spara registreringsnumret för bilar. Vad kan man säga om registreringsnumret? Jo, varje bil måste ha ett, och det skall vara unikt, det vill säga ingen bil får ha samma registreringsnumer som någon annan. Registreringsnumret används ju för att identifiera en bil så det borde passa bra att ha det som **primärnyckel**.

Vi kan uppdatera fältet med kommandot ”ALTER TABLE”:

```
mysql> ALTER TABLE bilar MODIFY reg char(10) PRIMARY KEY;
Query OK, 10 rows affected (0.29 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> EXPLAIN bilar;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| reg       | char(10) |      | PRI |         |       |
| marke     | char(50) | YES  |     | NULL    |       |
| modell    | char(50) | YES  |     | NULL    |       |
| arsmodell | int(11)  | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql>
```

Vi har nu sagt att fältet ”reg” skall vara primärnyckel. Eftersom primärnyckel automatiskt innebär att fältet inte får innehålla dubletter så är det ju det vi vill. Vi hade naturligtvis kunnat skapa fältet på detta sätt redan från början.

Fältet pris

Vi skall också lägga till ett fält till vår biltabell, nämligen pris. Vi kan säga att vi skall göra en liten tjänst åt en bilmäklare. Han vill ha en lista över bilar och vill då naturligtvis veta vad säljarna vill ha för sina bilar.

Än en gång får vi använda ”ALTER TABLE” men nu skall vi lägga till ett fält. Fältet skall heta ”pris” och vara av typen ”INT”, heltal.

```
mysql> ALTER TABLE bilar ADD pris INT;
```

Kontrollera att det blev som du tror.

Uppdatera poster med UPDATE

Nu har tabellen utökats med fältet ”pris” Det fältet är nu tomt för alla bilar, vi måste fylla det med något. Vi måste alltså ändra prosterna så att priset kommer med. Ändrar poster gör man med kommandot UPDATE, vi går igenom det lite kort här, det kommer igen i laboratoiner längre fram. **Varning! UPDATE förändrar data och det finns inget ”undo”.**

```
mysql> UPDATE bilar SET pris=130000 WHERE reg='ABC123';
```

Även här ser vi att vi väljer ut vilka fält som skall ändras med ”**WHERE**”, hade vi inte angett vilka rader som skall ändras skulle alla rader ha ändrats. Det vill säga alla bilar hade fått priset 130000, vilket förmodligen inte är vad vi vill. Uppdatera nu tabellen så att den ser ut så här.

```
mysql> SELECT * FROM bilar;
+--------+------------+-----------+-----------+--------+
| reg    | marke      | modell    | arsmodell | pris   |
+--------+------------+-----------+-----------+--------+
| ABC123 | Saab       | 9-5       |      2003 | 130000 |
| DEF123 | Volvo      | S80       |      2002 | 140000 |
| GHI123 | Mazda      | 626       |      2001 |  80000 |
| JKL123 | Audi       | A8        |      2001 | 150000 |
| MNO123 | BMW        | 323       |      1998 |  60000 |
| PQR123 | Ford       | Mondeo    |      2001 |  90000 |
| STU123 | Volvo      | 740       |      1987 |  35000 |
| VYX123 | Volkswagen | Golf      |      1988 |  40000 |
| ABC456 | Volkswagen | Polo      |      2003 |  75000 |
| DEF456 | Toyota     | Carina II |      1998 |  30000 |
+--------+------------+-----------+-----------+--------+
10 rows in set (0.01 sec)

mysql>
```

Ställa frågor med SELECT

Nu har vi en tabell med bilar som verkar fungera. Nu kan det vara läge att testa att välja saker ut den med ”**SELECT**”. Select har följande syntax (i sin enklaste form).

```
SELECT <vad> FROM <var>;
```

Allstå, välj någonting från någon tabell eller några tabeller. Lite exempel:

```
mysql> SELECT * FROM bilar;
```

Väljer allt (* betyder allt), från bilar. Allt refererar till fält. Det vill säga välj alla fält från bilar.

```
mysql> SELECT marke, modell, arsmodell FROM bilar;
```

Väljer fälten marke, modell och arsmodell från bilar.

Select väljer normalt alla poster i de fält som anges. Vill man begränsa det för man göra det med WHERE. Om man sätter ihop SELECT med WHERE så kan det se ut så här:

```
SELECT <vad> FROM <var> WHERE <villkor>;
```

Allstå , välj någonting från någon tabell eller några tabeller där ett villkor är uppfyllt. Vi tar ett exempel:

```
mysql> SELECT * FROM bilar WHERE marke='volvo';
```

Väljer ut alla bilar av märket Volvo. Vi kanske bara vill ha en lista med registreringsnumer för alla Volvobilar:

```
mysql> SELECT reg FROM bilar WHERE marke='volvo';
```

Enkelt vad? Man kan även göra mer avancerade saker. Säg att vi vill välja ut alla bilar som är från 2000 eller nyare:

```
mysql> SELECT * FROM bilar WHERE arsmodell>=2000;
```

På samma sätt för alla bilar som är nyare än 2000.

```
mysql> SELECT * FROM bilar WHERE arsmodell<2000;
```

Sortera med ORDER BY

Ofta vill man sortera eller gruppera sina resultat. Sorterar gör man med ”**ORDER BY**”. För att till exempel sortera efter årsmodell kan vi skriva.

```
mysql> SELECT * FROM bilar ORDER BY arsmodell;
```

Nu kommer vi att sortera efter årsmodell. Men vi kanske vill ha de nyaste bilarna först och inte de äldsta som det blir nu. Det är enkelt ordnat. Man kan lägga till **DESC** efter det fält som man skall sortera på för att sorterra i omvänd ordning. Desc är en förkortning för engelskans descending som betyder ”minskande”. Vi testar:

```
mysql> SELECT * FROM bilar ORDER BY arsmodell DESC;
```

Om vi nu har 30000 bilar varav 1000 är av årsmodell 2002 så kan det vara svårt att hitta bland dem i alla fall. Vi skulle vila sortera först på årsmodell och sedan på bilmärke för att göra det lättare. Det går naturligtvis. Man kan lägga till hur många fält som helst att sortera efter. Man skiljer dem åt med komma (,). Tänk på att eventuella DESC hör till ett sorteringsfält. För att sortera efter årsmodell i omvänd ordning och efter bilmärke i andra hand och modell i tredje hand skriver vi så här:

```
mysql> SELECT * FROM bilar ORDER BY arsmodell DESC, marke, modell;
```

Räkna med COUNT() och SUM()

COUNT()

Det kanske inte är intressant med en lista på alla bilar nyare än 2000. Har man 30000 bilar i sin tabell kanske man snarare är intresserad av hur många bilar som är nyare än 2000. För att få reda på det kan man använda funktionen COUNT() som räknar antalet poster som valts ut. Se nedanstående exempel:

```
mysql> SELECT COUNT(*) FROM bilar WHERE arsmodell>2000;
```

Stjärnan i COUNT(*) betyder att alla poster skall tas med. Skriver man istället COUNT(pris) så räknas alla poster där fältet pris har ett värde.

GROUP BY

Ibland vill man göra operationer, till exempel COUNT(), på olika grupper av poster. Då kan man använda **GROUP BY**. Group by är lite speciellt. Det används bara i samband med andra funktioner som till exempel COUNT(). I stället för att returnera posterna var för sig så returnernas de i grupper för att till exempel kunna räknas. Antag att du vill veta hur många bilar

det finns av varje årsmodell. Kör och fundera över följande sats:

```
mysql> SELECT arsmodell,COUNT(*) FROM bilar GROUP BY arsmodell;
```

SUM()

SUM() är en funktion som används för att summera fält. Vill man till exempel se hur mycket alla bilana är värda tillsammans kan man gör så här:

```
mysql> SELECT sum(pris) FROM bilar;
```

Man kan också får reda på hur mycket det sammanlagda värdet på bilarna av varje märke.

```
mysql> SELECT marke,SUM(pris) FROM bilar GROUP BY marke;
```

Det finns andra funktioner som fungerar på ungefär samma sätt. Till exempel **MIN()**, **MAX()** och **AVG()**. Min och max returnerar det högsta respektive lägsta värdet på ett fält. Avg är en förkortning för engelskans ”average” som betyder ”medelvärde”. På så sätt kan vi få reda på vilken bil som är dyrast respektive billigast och vi kan ta reda på medelpriset. Vi kan också kombinera dessa på alla tänkbara sätt.

Att svara på

Följande frågor skall du svara på. Visa eller lämna in dessa till laborationshandledaren.

Uppgift 1:

Skapa en fråga som väljer ut alla bilar av märket ”volvo” som från 2001 och nyare.

Svar:_____

Uppgift 2:

Skapa en fråga som visar hur många bilar det finns av varje märke.

Svar:_____

Uppgift 3:

Skapa en fråga som, per bilmodell skriver ut det högsta, lägsta och medelpriset. Tabellen skall se ut så här:

```
+------------+-----------+-----------+-----------+
| marke      | MIN(pris) | MAX(pris) | SUM(pris) |
+------------+-----------+-----------+-----------+
| Audi       |    150000 |    150000 |    150000 |
| BMW        |     60000 |     60000 |     60000 |
| Ford       |     90000 |     90000 |     90000 |
| Mazda      |     80000 |     80000 |     80000 |
| Saab       |    130000 |    130000 |    130000 |
| Toyota     |     30000 |     30000 |     30000 |
| Volkswagen |     40000 |     75000 |    115000 |
| Volvo      |     35000 |    140000 |    175000 |
+------------+-----------+-----------+-----------+
```

Svar:_____

Uppgift 4

Skapa en tabell som per årsmodell skriver ut hur många bilar det finns och deras medelpris. Svaret skall se ut så här:

```
+-----------+----------+-------------+
| arsmodell | COUNT(*) | AVG(pris)   |
+-----------+----------+-------------+
|      2003 |        2 | 102500.0000 |
|      2002 |        1 | 140000.0000 |
|      2001 |        3 | 106666.6667 |
|      1998 |        2 |  45000.0000 |
|      1988 |        1 |  40000.0000 |
|      1987 |        1 |  35000.0000 |
+-----------+----------+-------------+
```

Svar:_____