Laboration 3, Databashantering med MySQL

Av: Marcus Rejås <marcus@rejas.se>

> *I denna laboration skall vi jobba vidare på bildatabasen som vi började på förra gången. Denna gång skall vi skapa ytterligare en tabell och skapa relationer mellan dem.*
>
> *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) |      | PRI |         |       |
| marke     | char(50) | YES  |     | NULL    |       |
| modell    | char(50) | YES  |     | NULL    |       |
| arsmodell | int(11)  | YES  |     | NULL    |       |
| pris      | int(11)  | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
5 rows in set (0.05 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 ”agare”

Nu vill vi att man skall kunna lägga till en ägare på varje bil. En bil skall ha bara en enda ägare men en person skall kunna äga flera bilar. För att följa de normaliseringsregler[1] vi lärt oss så vet vi att vi måste skapa en tabell till med ägarna till bilarna. Det gör vi av i huvudsak för att förhindra dubbellagring eller redundans som det kallas. Man kan enkelt se att det behövs eftersom **ägaren inte beror helt på bilens nyckel**.

Vi börjar med att uppdatera biltabellen. Lägg till fältet agare, se förra labben om du inte minns hur man gör. Fältet skall ha typen **int**. Int är en förkortning för **integer** som betyder ”heltal”. När du är klar skall resultatet bli så här:

```
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    |       |
| pris      | int(11)  | YES  |     | NULL    |       |
| agare     | int(11)  | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

1 Se http://www.jonasbjork.net/databas/normalisering.html

Skapa ägare till bilarna

Nu skall dessa bilar få ägare. Det gör vi genom att mata in heltal i fältet agare för bilarna. Titta på din tidigare labb och lathunden om du inte minns hur man gör. **Kom ihåg att UPDATE är farligt**, glöm inte **WHERE.** När du är klar skall biltabellen se ut så här:

```
mysql> SELECT * FROM bilar;
+--------+------------+-----------+-----------+--------+-------+
| reg    | marke      | modell    | arsmodell | pris   | agare |
+--------+------------+-----------+-----------+--------+-------+
| ABC123 | Saab       | 9-5       |      2003 | 130000 |     1 |
| DEF123 | Volvo      | S80       |      2002 | 140000 |     2 |
| GHI123 | Mazda      | 626       |      2001 |  80000 |     3 |
| JKL123 | Audi       | A8        |      2001 | 150000 |     5 |
| MNO123 | BMW        | 323       |      2001 |  60000 |     5 |
| PQR123 | Ford       | Mondeo    |      2001 |  90000 |     4 |
| STU123 | Volvo      | 740       |      1987 |  35000 |     5 |
| VYX123 | Volkswagen | Golf      |      1983 |  40000 |     5 |
| ABC456 | Volkswagen | Polo      |      2003 |  75000 |     1 |
| DEF456 | Toyota     | Carina II |      1998 |  30000 |     2 |
+--------+------------+-----------+-----------+--------+-------+
11 rows in set (0.01 sec)
```

Från början har alla poster värdet **NULL** i fältet ägare (man kan ange ett annat defaultvärde när man skapar ett fält om man vill). NULL är ett speciellt värde som betyder ”ingenting”. Lägg nu till en bil av valfritt märke och modell som inte har någon ägare, det vill säga, låt agare vara NULL.

Tabellen personer

Vi skall nu skapa en tabell med personer. Tabellen skall naturligtvis hålla poster med information om de olika personerna. Varje person skall identifieras av ett id-nummer. Om vi vore en förening kanske detta skulle kallas medlemsnummer, vore vi ett företag kanske det skulle kallas anställningsnummer. Vi använder ett löpnummer istället för personernas personnummer eftersom det är bra för personerna att hålla sitt personnummer för sig själv.

Den information vi skall spara om varje person är, förutom id-numret, förnamn och efternamn. Vi kallar dessa fält för ”fnamn” och ”enamn”. Fältet med id-numret kallar vi för id. Eftersom detta fält skall unikt identifiera en person så gör vi detta till **primärnyckel** i tabellen. Vi sätter det också till att räkna upp sig självt (AUTO_INCREMENT) så slipper vi tänka på det. Skapa nu tabellen. Den skall se ut så här när den är klar (titta i föregående labb om du inte vet hur du skall göra):

```
mysql> DESCRIBE personer;
+-------+----------+------+-----+---------+----------------+
| Field | Type     | Null | Key | Default | Extra          |
+-------+----------+------+-----+---------+----------------+
| id    | int(11)  |      | PRI | NULL    | auto_increment |
| fnamn | char(50) | YES  |     | NULL    |                |
| enamn | char(50) | YES  |     | NULL    |                |
+-------+----------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM personer;
+----+-----------+-------+
| id | fnamn     | enamn |
+----+-----------+-------+
|  1 | Kalle     | Anka  |
|  2 | Kajsa     | Anka  |
|  3 | Knatte    | Anka  |
|  4 | Tjatte    | Anka  |
|  5 | Fnatte    | Anka  |
|  6 | Knase     | Anka  |
```



```
|  7 | Alexander | Lukas |
+----+-----------+-------+
7 rows in set (0.03 sec)
```

Nu har vi lite frågor som vi kan ställa frågor till. Vi skall nu öva oss på att använda **SELECT;**

SELECT från flera tabeller

Repetera SELECT från förra laborationen om du känner dig osäker på hur det fungerar.

I den förra laborationen använde vi SELECT för att välja saker från en tabell. Nu skall vi utveckla detta vidare och välja från flera olika tabeller.

Vi börjar med en liten fråga som listar alla bilar och deras ägare. Nu skall vi i SELECT välja inte bara från en tabell utan från flera (två i detta fall). Eftersom de fält vi väljer kan ha samma namn olika tabeller måste vi ange både vila fält vi skall välja och i vilka tabeller dessa kan hittas. Vill vi välja fnamn från person så anges detta som ”person.fnamn”. Vill vi välja alla fält från person så kan vi skriva person.*. Som vanligt åtskiljer vi det vi listar med kommatecken. Även tabellerna måste anges och åtskiljs med kommatecken. Vi börjar med att välja bilmärken och modeller och så förnamn och efternamn ur persontabellen. Vi skriver så här:

```
mysql> SELECT bilar.marke,bilar.modell,personer.fnamn,personer.enamn
FROM personer, bilar;
```

Men det där blev ju inte så bra! Varför inte det? Jo vi har ingenstans angivit hur tabellerna skall relatera till varandra. Det kan vi göra med hjälp av kommandot **INNER JOIN**. Tag dig tid att fundera på nedanstående frågor då dessa till en början kan verka krångliga.

INNER JOIN

INNER JOIN fungerar så att du väljer fält från flera tabeller. Men du anger att du skall välja från bara en tabell och sedan koppla ihop dessa med en relation. Detta kan du göra med INNER JOIN. För att åstadkomma det som vi ville göra ovan kan man skriva så här:

```
mysql> SELECT bilar.marke,bilar.modell,personer.fnamn,personer.enamn
FROM bilar INNER JOIN personer ON bilar.agare=personer.id;
+--------------+-----------+--------+-------+
| marke        | modell    | fnamn  | enamn |
+--------------+-----------+--------+-------+
| Saab         | 9-5       | Kalle  | Anka  |
| Volkswagen   | Polo      | Kalle  | Anka  |
| Volvo        | S80       | Kajsa  | Anka  |
| Toyota       | Carina II | Kajsa  | Anka  |
| Mazda        | 626       | Knatte | Anka  |
| Ford         | Mondeo    | Fnatte | Anka  |
| Audi         | A8        | Tjatte | Anka  |
| BMW          | 323       | Tjatte | Anka  |
| Volvo        | 740       | Tjatte | Anka  |
| Volkswagen   | Golf      | Tjatte | Anka  |
+--------------+-----------+--------+-------+
11 rows in set (0.00 sec)

mysql>
```

Eftesom vi inte anger någon sorteringsordning (... ORDER BY ...) så kan vi inte säga något om ordningen på resultatet. Du kan alltså få en lista i en annan ordning, men rätt ägare skall naturligtvis stå vid rätt bil.

Denna fråga kan man läsa ut så här: ”Välj fälten märke från bilar, modell från bilar, förnamn från peroner och efternamn från personer från bilar ihopslaget med personer där ägare i bilar är samma som id i personer. INNER JOIN tar bara med poster där kopplingen stämmer. Personer som inte äger några bilar eller bilar som inte har någon ägare kommer inte med om man använder INNER JOIN.

Vi tar ett exempel till. Vi vill välja ut samma sak som ovan men utgå från personerna.

```
mysql> SELECT bilar.marke,bilar.modell,personer.fnamn,personer.enamn
FROM personer INNER JOIN bilar ON bilar.agare=personer.id;
```

Och som vi ser så går det precis lika bra!

OUTER JOIN

Som vi såg så tog INNER JOIN bara med de poster där kopplingen stämmer. Ibland kanske man vill ha med även de poster från någon tabell som inte är med i kopplingen. I vårt exempel kanske man vill lista alla bilar, även de som inte ägs av någon eller alla personer, även de som inte har någon bil. Till exempel om denna databas är ett medlemsregister på den bilklubb där man kan vara medlem även utan att äga en bil. Då vill man kanske ta ut en lista på alla medlemmar och för de medlemmar som har en bil, även deras bil. Då använder man OUTER JOIN.

OUTER JOIN tar med allt från en tabell och de rader som passar i den andra. För att man skall veta vilken av tabellerna som allt skall tas med ifrån så finns det två olika OUTER JOIN. Dessa heter ”LEFT OUTER JOIN ... ON” och ”RIGHT OUTER JOIN ... ON”.

Dessa fungerar på följande sätt:

```
SELECT * FROM bilar RIGHT OUTER JOIN personer ON bilar.agare=personer.id;
```

Titta på den del av uttrycket som är ” **bilar RIGHT OUTER JOIN personer**”. Vi har en **RIGHT** OUTER JOIN vilket betyder att allt i tabeller till **höger** (alltså personer) skall tas med.

Att svara på

Följande frågor skall du svara på. Visa eller lämna in dessa till laborationshandledaren. Naturligtvis provkör du frågorna i din databas innan du skriver dem på detta blad.

Uppgift 1:

Det är dags att betala bilskatt. Skapa en fråga som listar alla personer **som har en bil** och registreringsnumret på deras bil.

Svar:_____ _____

Uppgift 2:

Skapa en fråga som visar **alla bilar** och deras ägare om de har någon.

Svar:_____ _____

Uppgift 3:

Skapa en fråga som visar **alla personer** och deras bil om de har någon.

Svar:_____ _____

Uppgift 4

Skapa en fråga som visar förnamn och efternamn på alla som har en Saab.

Svar:_____ _____

Uppgift 5[*]

Skapa en fråga som listar namnen på alla personer som har en eller flera bilar, samt hur många bilar de har.

Svar:_____ _____

* Uppgift som är lite svårare

