

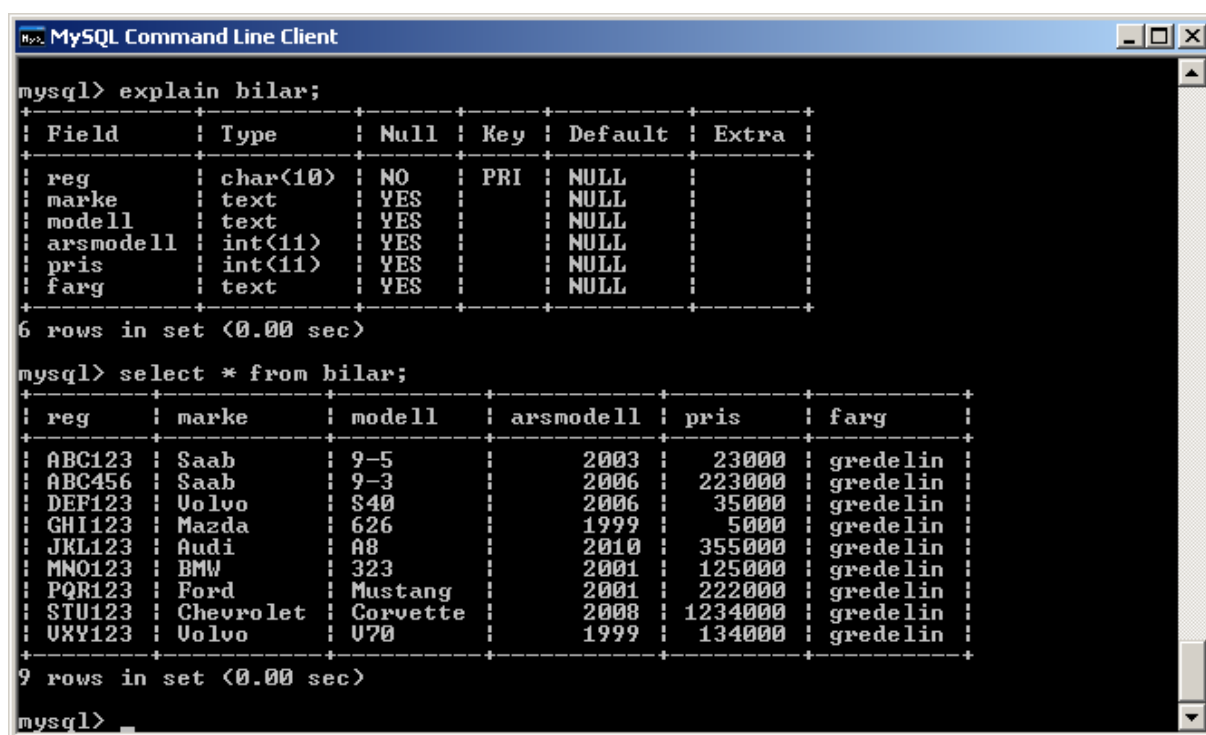
Övning 8 – SQL grunderna III

Övning 10 är det tredje steget i vår utforskning av SQL med hjälp av MySQL! Som vanligt är det viktigt att du har gjort och förstått de tidigare stegen för att gå vidare. Ha tidigare övningar tillgängliga så du kan titta där vid behov!

I denna övning skall vi arbeta med två olika tabeller!

Innan du är klar med övningen skall du för mig ha visat upp dina svar på de sista frågorna i övningen!

- ✓ Starta MySQL och ta fram databasen "fordon" från förra övningen! Så här såg tabellen bilar ut:



```
mysql> explain bilar;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| reg   | char(10) | NO | PRI | NULL | |
| marke | text | YES | | NULL | |
| modell | text | YES | | NULL | |
| arsmode | int(11) | YES | | NULL | |
| pris  | int(11) | YES | | NULL | |
| farg  | text | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

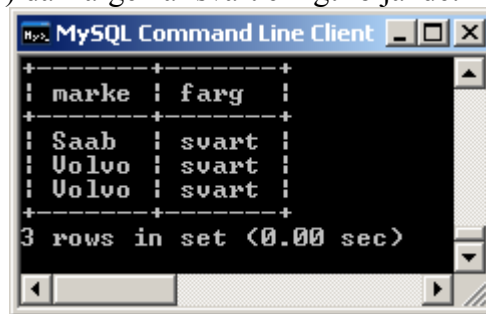
mysql> select * from bilar;
+-----+-----+-----+-----+-----+-----+
| reg   | marke | modell | arsmode | pris  | farg  |
+-----+-----+-----+-----+-----+-----+
| ABC123 | Saab | 9-5 | 2003 | 23000 | gredelin |
| ABC456 | Saab | 9-3 | 2006 | 223000 | gredelin |
| DEF123 | Volvo | S40 | 2006 | 35000 | gredelin |
| GHI123 | Mazda | 626 | 1999 | 5000 | gredelin |
| JKL123 | Audi | A8 | 2010 | 355000 | gredelin |
| MNO123 | BMW | 323 | 2001 | 125000 | gredelin |
| PQR123 | Ford | Mustang | 2001 | 222000 | gredelin |
| STU123 | Chevrolet | Corvette | 2008 | 1234000 | gredelin |
| VXY123 | Volvo | U70 | 1999 | 134000 | gredelin |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)

mysql>
```

Först lite repetition från förra övningen! Försök svara på följande frågor:

1. Alla Volvobilar är svarta. Ange en enda SQL-fråga för att ändra deras färg till svart!
2. Sätt även Saabs 9-5 till svart!

3. Visa alla bilar (poster) där färgen är svart enligt följande:

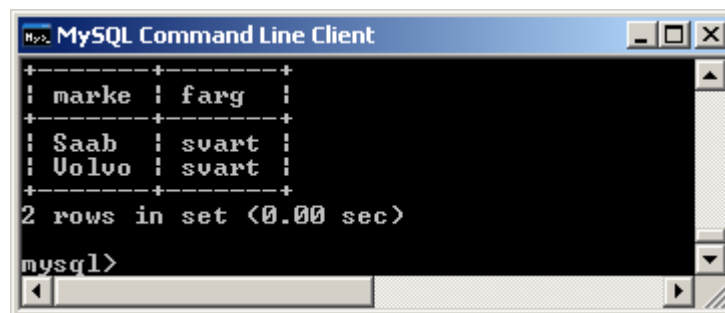


```
mysql> SELECT marke, farg FROM bilar WHERE farg = 'svart';
```

marke	farg
Saab	svart
Volvo	svart
Volvo	svart

3 rows in set (0.00 sec)

4. Visa alla bilmärken där minst en bil är svart (dvs vi är inte intresserade att lista varenda enskild bil, bara bilmärket!):



```
mysql> SELECT marke FROM bilar WHERE farg = 'svart' GROUP BY marke;
```

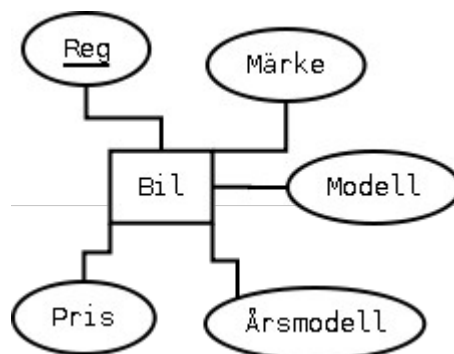
marke
Saab
Volvo

2 rows in set (0.00 sec)

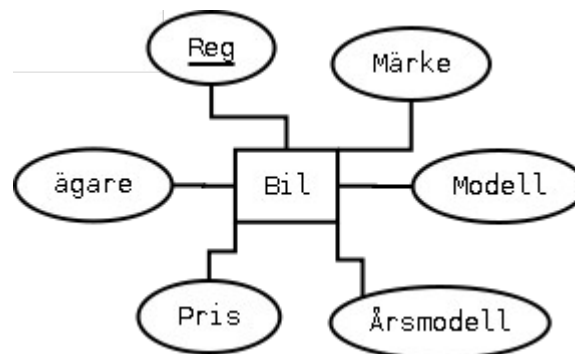
mysql>

5. Ta bort fältet "färg"!

Stanna nu upp ett ögonblick och plocka fram dina kunskaper om ER-modellering! Då skulle SQL-tabellen "bilar" kunna ER-modelleras så här, där alla fält i tabellen utgör attribut, och registreringsnumret ("reg") är en nyckel:

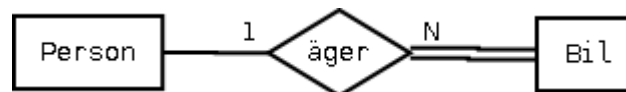


Antag att vi vill föra in uppgiften i vår databas att varje bil har en ägare. Hur skulle vi göra det med hjälp av ER-modelleringen? Kanske så här...?



Hmmm...det här känns onaturligt! En ägare är ju (oftast) en person, som beskrivs av sina egna attribut (personnr, namn, adress,...). Nej, "ägare" hör inte hemma som en beskrivning på en bil - "ägare" är alldeles för viktig för att bara vara ett attribut till entiteten "bil"!

Det naturliga är att göra ägaren till en "person" som kan "äga" en bil (eller ingen eller flera):



Enligt ovanstående ER-diagram (attributen ej utskrivna) måste också en bil ägas av en enda person, vilket ju ofta är fallet (vi bortser från att en bil kan ägas av ett företag).

Entiteten "Bil" kan då motsvaras av vår SQL-tabell "bilar". På motsvarande sätt borde alltså en "Person" motsvaras av en SQL-tabell "personer". Vi har då uppfyllt den s.k. "andra normalformen" (2NF), som säger att fält i en tabell måste på något sätt ha samband med den använda nyckeln. Med 2NF undviker man bl.a. att spara samma data flera gånger. Men mer om normalformer lite senare¹!

Du skall nu alltså skapa en tabell "personer". Tabellen skall innehålla 3 fält: id-nummer, förnamn och efternamn. ID-numret kan motsvara ett medlemsnr i en organisation, anställningsnr i ett företag etc. I Sverige tycker man om att använda personnr till allting, men det bör undvikas (personlig integritet!). Id-numret skall används som primärnyckel.

I SQL finns det en funktion (Auto_INCREMENT) som automatiskt räknar upp data med ett varje gång data skall läggas till. Praktiskt, så det kommer vi att använda oss av!

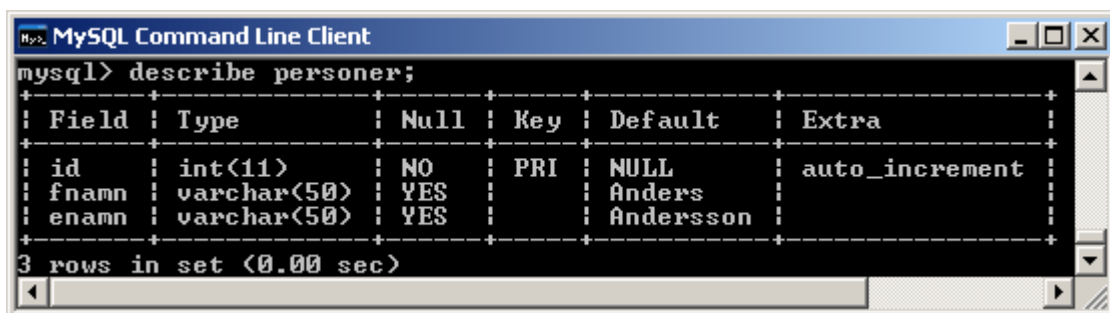
¹Är du nyfiken kan du alltid titta i Björk, kapitel 6!

- ✓ Skapa tabellen "personer" genom att ge kommandot
"create table personer(id int auto_increment primary key,
fnamn varchar(50),
enamn varchar(50) default 'Andersson');"

Id-numret blir alltså ett heltal (int), primärnyckel och räknas upp automatiskt. Datatypen varchar(50) (variable character) innebär att data får bestå av maximalt 50 tecken – består förnamnet av färre tecken så tar då förnamnet mindre plats i datorn (datatypen char(50) innebär att förnamnet alltid tar upp 50 tecken, oavsett om förnamnet t.ex. är "Bo"!).

När man sätter in data i tabellen så blir efternamnet automatiskt (=default) Andersson om man inte anger något annat. Man kan också lägga till ett default värde i en redan skapad tabell.

- ✓ Lägg till ett default värde för fältet "fnamn" genom att skriva
"alter table personer alter column fnamn set default 'Anders';"
Kontrollera att det blir rätt!



```
mysql> describe personer;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
fnamn	varchar(50)	YES		Anders	
enamn	varchar(50)	YES		Andersson	

3 rows in set (0.00 sec)

6.Ändra default-värdet för efternamnet till Svensson!

7.Lägg nu till ett antal namn (kom ihåg att id räknas upp automatiskt!), så att tabellen "personer" ser ut så här:



```
mysql> select * from personer;
```

id	fnamn	enamn
1	Darth	Uader
2	Kent	Smurf
3	Pippi	Långstrump
4	Anna	Anka
5	Mr	Spock

5 rows in set (0.00 sec)

Titta igen på fältet "id" - t.ex. medlemsnummer kan bestå av 8 siffror, och då vill man att alla 8 siffror alltid skall visas, ev med nollor framför! Det kan vi fixa!

- ✓ Ändra fältet "id" så att 8 siffror alltid visas - ange SQL-frågan "alter table personer modify id int(8) zerofill ;"

```
mysql> explain personer;
+-----+-----+-----+-----+-----+-----+
| Field | Type                | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(8) unsigned zerofill | NO   | PRI | 00000000 |       |
| fnamn | varchar(50)          | YES  |     | Anders  |       |
| enamn | varchar(50)          | YES  |     | Svensson |      |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

mysql> select * from personer;
+-----+-----+-----+
| id    | fnamn | enamn |
+-----+-----+-----+
| 00000001 | Darth | Vader |
| 00000002 | Kent  | Smurf |
| 00000003 | Pippi | Långstrump |
| 00000004 | Anna  | Anka  |
| 00000005 | Mr    | Spock |
+-----+-----+-----+
5 rows in set (0.00 sec)

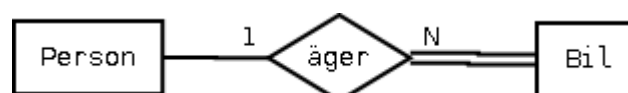
mysql>
```

Notera att åttan i int(8) endast anger antal siffror som skall visas! Datatypen int (heltal) lagras alltid i 4 bytes, oavsett storleken. Vilket innebär att int(8) och int(11) båda kan lagra lika stora tal! Detta skiljer sig från textbaserade datatyper, där t.ex. char(25) innebär att man kan lägga 25 tecken!

Notera också att zerofill automatiskt gjorde så att heltalen blev "unsigned" vilket innebär att inga negativa tal kan sparas (tal mellan 0 – 4294967295 kan lagras!)

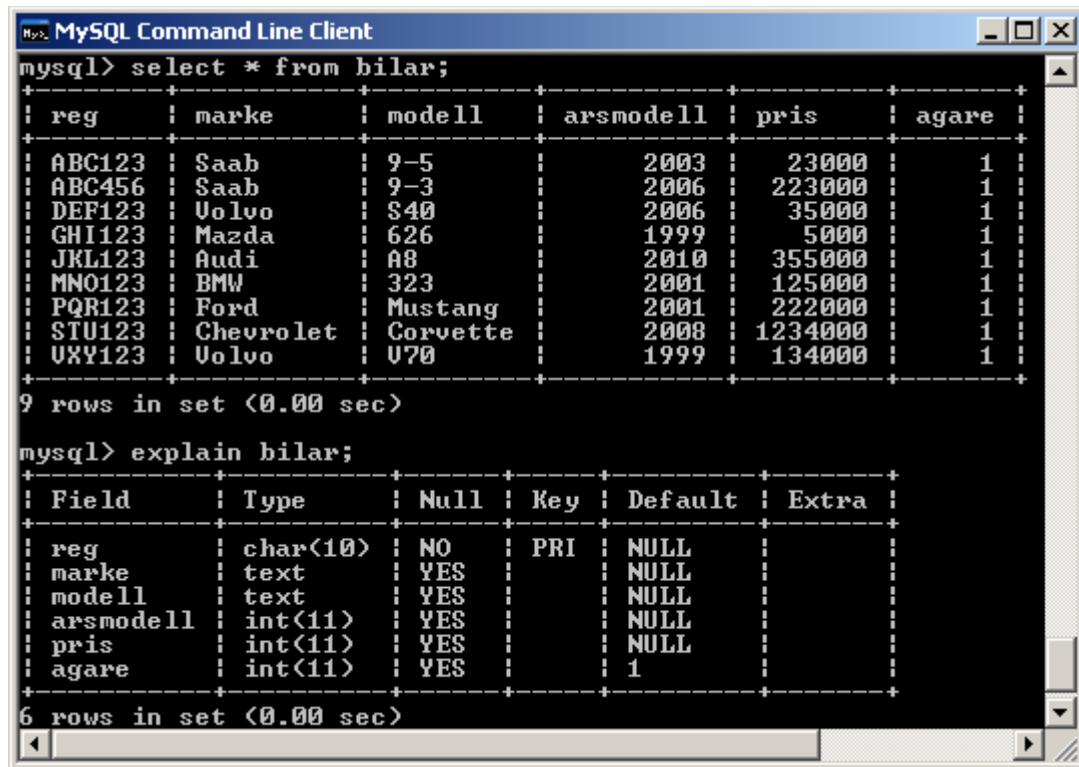
Vill man att auto_increment skall börja med en annan siffra än 1 (t.ex. 123), så kan man skriva "alter table personer auto_increment = 123;" Observera att detta gäller hela tabellen!

Ok, nu finns det två tabeller, en med bilar och en med personer. Men hur kopplas de ihop, dvs hur visar man sambandet att en person kan äga en bil?



Jo, man lägger till en kolumn i tabellen "bilar" som "pekar" på rätt person i tabellen "personer"!

8. Lägg till ett fält "ägare" i tabellen "bilar"! OBS! I fältet skall det stå data av samma datatyp som fältet "id" i tabellen "personer", och defaultvärdet skall vara 1! "bilar" skall då se ut så här:



The screenshot shows the MySQL Command Line Client interface. The first query is `select * from bilar;`, which returns 9 rows of car data. The second query is `explain bilar;`, which shows the table's structure with 6 columns: reg, marke, modell, arsmode11, pris, and agare.

reg	marke	modell	arsmode11	pris	agare
ABC123	Saab	9-5	2003	23000	1
ABC456	Saab	9-3	2006	223000	1
DEF123	Volvo	S40	2006	35000	1
GHI123	Mazda	626	1999	5000	1
JKL123	Audi	A8	2010	355000	1
MNO123	BMW	323	2001	125000	1
PQR123	Ford	Mustang	2001	222000	1
STU123	Chevrolet	Corvette	2008	1234000	1
VXY123	Volvo	U70	1999	134000	1

Field	Type	Null	Key	Default	Extra
reg	char(10)	NO	PRI	NULL	
marke	text	YES		NULL	
modell	text	YES		NULL	
arsmode11	int(11)	YES		NULL	
pris	int(11)	YES		NULL	
agare	int(11)	YES		1	

9. Varför vill vi ha ett default-värde för fältet "ägare"? (tips, kolla ER-diagrammet!)
Varför skulle vi INTE vilja ha default-värde?

10. Vem äger alltså alla bilarna "default"?

*Nej, nu måste du ändra i databasen så att bilarna ägs av **olika** personer!*

11. Ändra nu så att:

- Darth Vader äger Corvetten, Mustangen och Audi A8 (han bara roffar år sig...)
- Kent Smurf äger Mazda 626
- Pippi Långstrump äger de "svenska" bilarna (Pippi är världens svensk...)
- Anna Anka äger BMWn (är säkert en cabriolet, poppis i USA!)
- Mr Spock äger ingen bil alls (varför skulle han? Han åker ju på rymdskeppet Enterprise...)

12.Uppfyller nu din databas ER-modellen? Motivera!

13.Fältet "ägare" kopplar en specifik bil till en specifik ägare. Vad kallas fältet "ägare" då?

Jamen, det här var ju toppen! Tänk vad man kan få reda på från den här lilla databasen...hehe. Fast, vänta! Hur tar man i praktiken reda på vilka bilar Mr Spock äger (han äger inga...men det vet ju inte Mårten Gås, som är världens nyfiken på folks bilar och vill använda din databas!)? Jo, det skall vi kika på nu!

Eftersom vi har flera tabeller, så måste MySQL veta i vilken tabell ett speciellt fält ligger. Menar vi t.ex. fältet "fnamn" i tabellen "personer" så anger man det som "personer.fnamn". Om tabellernas fält inte har samma namn så behöver man i princip inte ange vilken tabell fältet kommer ifrån, men för tydlighetens skull så kan det ändå vara en bra idé!

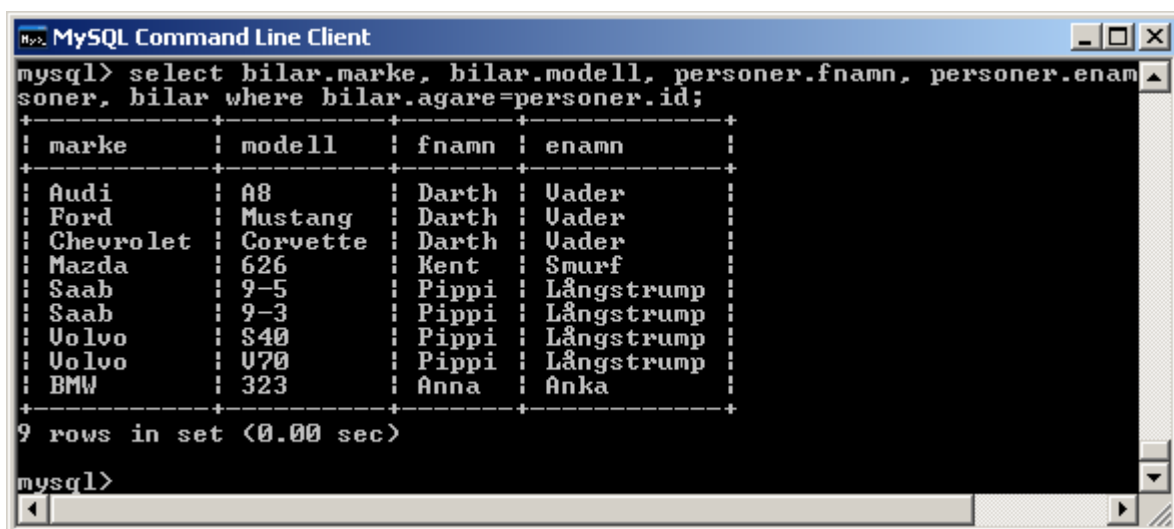
Nu vill vi lista ägarna till alla bilar!

- ✓ Ange SQL-frågan "select bilar.marke, bilar.modell, personer.fnamn, personer.enamn from person, bilar;"

Ja det var ingen höjdare! Alla bilar listas fem gånger med alla namnen!

Problemet var att vi inte angett HUR tabellerna skall relatera till varandra, så MySQL antag bara att vi vill lista alla möjliga kombinationer av bilar och namn!

- ✓ Ange SQL-frågan "select bilar.marke, bilar.modell, personer.fnamn, personer.enamn from person,bilar where bilar.agare=personer.id;"



```
mysql> select bilar.marke, bilar.modell, personer.fnamn, personer.enamn
from person, bilar where bilar.agare=personer.id;
```

marke	modell	fnamn	enamn
Audi	A8	Darth	Uader
Ford	Mustang	Darth	Uader
Chevrolet	Corvette	Darth	Uader
Mazda	626	Kent	Smurf
Saab	9-5	Pippi	Långstrump
Saab	9-3	Pippi	Långstrump
Volvo	S40	Pippi	Långstrump
Volvo	U70	Pippi	Långstrump
BMW	323	Anna	Anka

```
9 rows in set (0.00 sec)

mysql>
```

Här kopplar man alltså samman tabellerna med ett villkor – att "agare" och "id" skall innehålla samma sak! Man kan se det på så sätt, att från alla möjliga kombinationer av poster från de två tabellerna (det resultat du fick innan detta) så tar man ut de kombinationer där agare=id!

Man kan få samma resultat genom att ange SQL-frågan på ett annat sätt. Man kopplar då ihop tabellen "bilar" med tabellen "personer" genom att ange "inner join" (man 'joinar' tabellerna, slår ihop dem):

- ✓ Ange SQL-frågan **"select bilar.marke, bilar.modell, personer.fnamn, personer.enamn from bilar inner join personer on bilar.agare=personer.id;"**

Ovan så utgick vi från bilarna. Man kan också utgå från personerna. Resultatet blir det samma:

- ✓ Ange SQL-frågan **"select bilar.marke, bilar.modell, personer.fnamn, personer.enamn from personer inner join bilar on bilar.agare=personer.id;"**

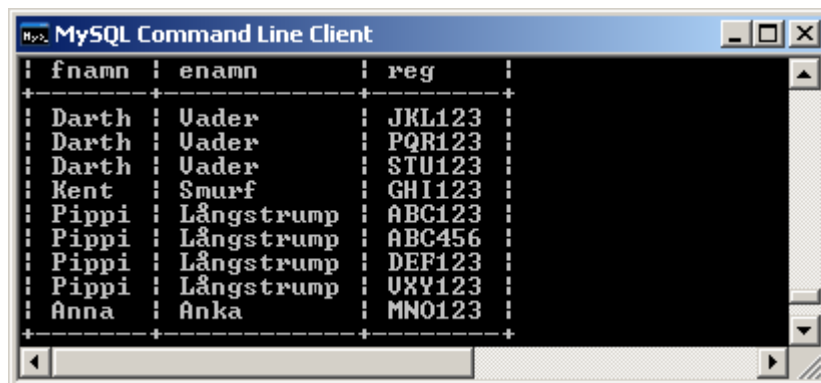
Kanske vill man sortera utskriften på annat sätt...

- ✓ Ange SQL-frågan **"select bilar.marke, bilar.modell, personer.fnamn, personer.enamn from personer inner join bilar on bilar.agare=personer.id order by marke desc;"**

GÖR FÖLJANDE UPPGIFTER OCH VISA UPP FÖR MIG:

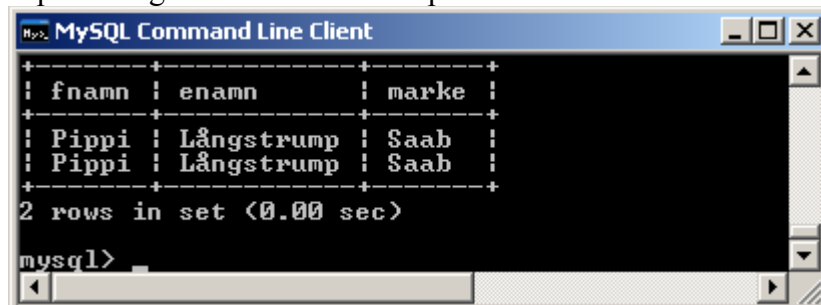
Uppgift 1: Visa hur du gjorde fråga 11 ovan!

Uppgift 2: Nu vill skatteverket veta vilka personer som äger bilar! Skapa en SQL-fråga som listar alla personer som har bil samt bilens (eller bilarnas) registreringsnummer!



fnamn	enamn	reg
Darth	Uader	JKL123
Darth	Uader	PQR123
Darth	Uader	STU123
Kent	Smurf	GHI123
Pippi	Långstrump	ABC123
Pippi	Långstrump	ABC456
Pippi	Långstrump	DEF123
Pippi	Långstrump	UXY123
Anna	Anka	MNO123

Uppgift 3: Skapa en fråga som visar namnen på de som har Saab:

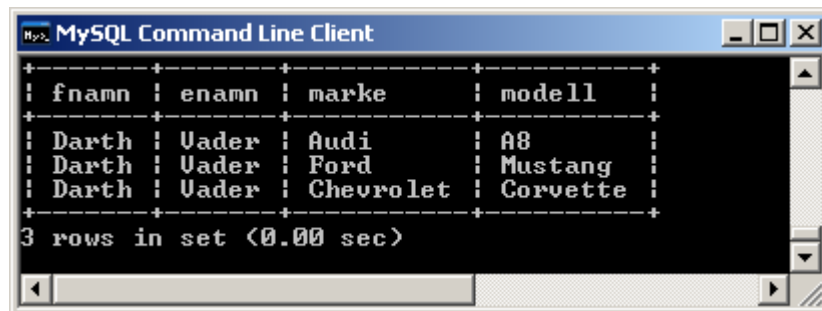


fnamn	enamn	marke
Pippi	Långstrump	Saab
Pippi	Långstrump	Saab

2 rows in set (0.00 sec)

mysql>

Uppgift 4: Skapa en fråga som listar vilka bilar Darth Vader har!



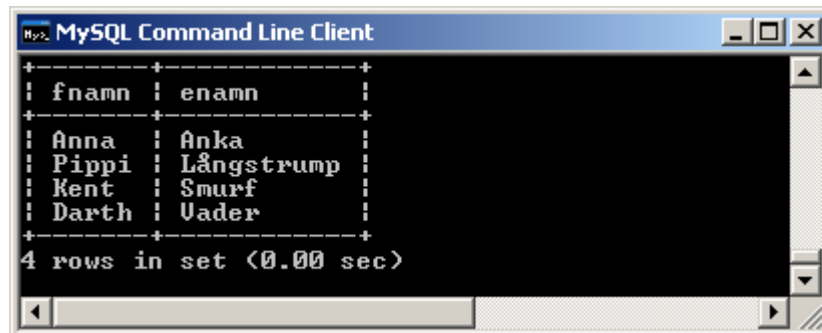
A screenshot of the MySQL Command Line Client window. The title bar says 'MySQL Command Line Client'. The window displays a table with four columns: 'fnamn', 'enamn', 'marke', and 'modell'. The table contains three rows of data, all for 'Darth Vader'. The status bar at the bottom indicates '3 rows in set (0.00 sec)'.

fnamn	enamn	marke	modell
Darth	Vader	Audi	A8
Darth	Vader	Ford	Mustang
Darth	Vader	Chevrolet	Corvette

Uppgift 5: Skapa nu en fråga som listar vilka bilar Mr Spock har!

Extrauppgifter (lite svårare):

Extrauppgift 1: Lista alla personer som har en bil, men skriv inte ut alla deras bilar!



A screenshot of the MySQL Command Line Client window. The title bar says 'MySQL Command Line Client'. The window displays a table with two columns: 'fnamn' and 'enamn'. The table contains four rows of data. The status bar at the bottom indicates '4 rows in set (0.00 sec)'.

fnamn	enamn
Anna	Anka
Pippi	Långstrump
Kent	Smurf
Darth	Vader

Extrauppgift 2: Lista alla personer som har en eller flera bilar, och ange hur många de har!