Inlämningsuppgift 2
Databassystem

Hannes Fahlin
a22hanfa

1. Verksamheten

Relationsdatamodellen är skapad utifrån att analysera ER-modellen. ER-modellen gav ett grafiskt perspektiv på hur dessa entiteter och attribut var kopplade. Relationsdatamodellen har skapat en abstrakt modell av data-relationer, data har organiserats och berättat relationer mellan olika data. Relationen har identifierat och definierat entiteter, attribut och relationer.

Entiteter i ER-modellen blev identiferade och därefter skapades det en motsvarande tabell i relationsdatamodellen. Därefter blev attributen kolumner i tabellen. Relationerna har etablerats genom att koppla samman olika främmande nycklar.

Semantik från ER-modellen till relationsmodellen och databastabellerna förlorades. I uppgiftsbeskrivningen behövdes en relationsdatamodell baserat på ER-modellen. Där modellen översattes. I ER-modellen finns det en många till många relation mellan funktionär och ammunition. För att representera denna relation behövs en separat tabell. Därför skapades tabellen ammuntionÄgs. Tabellen fungerar som kopplingstabell. Den är nödvändig för översättningen mellan ER och relationsmodellen.

2. Realisera Databasen

Skjutfält ($\underline{namn}$, tele, stad)
Skjutbana ($\underline{nr}$, moment, $\overline{skjutfält}$)
Skytt($\underline{pnr}$, namn, skyttlag)
Gevär($\underline{namn}$, vikt, $\overline{\underline{skyttPnr}}$)
Funktionär($\underline{pnr}$, lön, namn, $\overline{skjutbana}$, $\overline{fält}$)
Ammunition($\underline{kaliber}$, namn, $\overline{\underline{gevärNamn}}$, $\overline{\underline{skyttPr}}$)
Måltavla($\underline{nr}$, antal)
Skjutserie($\underline{starttid}$, resultat, $\overline{\underline{skyttPnr}}$, $\overline{bevakarPnr}$, $\overline{\underline{målNr}}$, $\overline{banNr}$)
ammuntionÄgs ($\overline{\underline{funkPnr}}$, $\overline{\underline{ammunitionNamn}}$, $\overline{\underline{skyttPnr}}$)

Databastabeller är den konkreta implementeringen av relationsmodellen. I en databas består en databastabell av rader och kolumner. Kolumner representerar attribut medan varje rad representerar instans av data.

Databasen skapas genom att först kolla om databasen med namnet redan existerar, om den existerar tas hela databasen bort. Därefter skapas antingen databasen på nytt eller skapar en helt ny. Sedan väljs databasen genom att specificera namnet på databasen.

```
DROP DATABASE IF EXISTS a22hanfa;
CREATE DATABASE a22hanfa;
USE a22hanfa;"
```

Tabellen skjutfält innehåller attribut, information sparas i tabellen om ”namn”, ”telefonnummer” och ”stad” för varje instans av skjutfält.

Namn är en VARCHAR. VARCHAR är en datatyp som passar till textsträngar, eftersom den kan spara information i flexibel längd. En flexibel längd är optimerat för mindre lagringsutrymme. Datatypen är satt till en maximal längd på 20 tecken, vilket ger tillräckligt med utrymme för namn. Namn är primärnyckeln för skjutfält-tabellen. Attributet Tele är av datatypen INT. Typen är passande till telefonnummer, eftersom datatypen INT sparar heltal. Stad använder sig också av en VARCHAR av samma anledning som Namn, det är en optimerad datatyp som passar bra strängar.

```
CREATE TABLE skjutfält(
   namn VARCHAR(40),
   tele VARCHAR(40),
   stad VARCHAR(40),
   PRIMARY KEY(namn)
)ENGINE=innodb;
```

Tabellen skjutbana skapas med attributen ”nr”, ”moment” och ”skjutfältNamn”. Nummer attributen nr representerar skjutbanans nummer i relation till andra skjutbanor, numret sparas i en VARCHAR datatyp, detta på grund av att första nollan tas bort i en INT datatyp, nr är primärnyckeln för skjutbana tabellen. Både skjutfältNamn och moment använder VARCHAR datatypen. Samma princip som tidigare är anledning till att vi använder oss av just den datatypen. VARCHAR sparar enbart den faktiska datamängden på strängen, detta sparar utrymme i databasen, jämförelsevis med CHAR som alltid kommer spara en specificerad längd karaktärer.

```
CREATE TABLE skjutbana(
   nr INT,
   moment VARCHAR(40),
   skjutfältNamn VARCHAR(40),
   PRIMARY KEY(nr),
   FOREIGN KEY(skjutfältNamn) REFERENCES skjutfält(namn)
)ENGINE=innodb;
```

I tabellen för skytt existerar attributen ”pnr”, ”namn” och ”skyttelag”. Personnummer attributet pnr är primärnyckeln. Personnumret sparas i variabeln pnr, datatypen CHAR används till att spara personnumret. Detta är eftersom personnumret inte bara innehåller heltal utan också ett bindestreck. Det kan maximalt sparas 13 karaktärer i variabeln, detta är på grund av att ett svenskt personnummer som kortast är 10 siffror och som längst 13 siffror, beroende på om ett bindestreck eller de två första siffrorna används. Personnumret är också indexerad, eftersom det är många andra tabeller som beror på numret, med index snabbas sökningshastigheten upp. Namn och skyttelag har VARCHAR som datatyp eftersom namn är

svåra att veta exakta längd på innan de läggs in, därmed kan det sparas i en variabel med varierande längd för att spara på lagringsutrymme.

```
CREATE TABLE skytt(
   pnr CHAR(13),
   namn VARCHAR(40),
   skyttelag VARCHAR(40),
   PRIMARY KEY(pnr),
   INDEX (pnr)
)ENGINE=innodb;
```

I tabellen gevär har vi tre kolumner, ”namn”, ”vikt” och ”skyttPnr”. Denna tabell har en sammansatt primärnyckel, namn och skyttPnr. Gevärets namn bestäms i en VARCHAR eftersom ett vapen kan innehålla både siffror och bokstäver, i varierande längder av strängar. Vikten av gevär är deklarerat i en DECIMAL eftersom vapen tenderar att väga olika mycket och förmodligen inte ett exakt heltal används datatypen. Datatypen är också specificerad att kunna ha ett värde mellan -999,99 till 999,99, detta eftersom det kan finnas tyngre vapen som är stationära medan mera mobila vapen finns. Namn är en VARCHAR av samma anledning som tidigare sagts. Anledningen till att skyttPnr också är en CHAR har nämnts tidigare. SkyttPnr är en främmandenyckel som hämtas från skytt-tabellen, detta är eftersom gevärets ägare behöver kunna identifieras. Namn och skyttPnr är indexerat, detta på grund av att de är saker som söks efter.

```
CREATE TABLE gevär(
   namn VARCHAR(40),
   vikt DECIMAL(5,2),
   skyttPnr CHAR(13),
   PRIMARY KEY(namn, skyttPnr),
   FOREIGN KEY(skyttPnr) REFERENCES skytt(pnr),
   INDEX (namn, skyttPnr)
)ENGINE=innodb;
```

Tabellen funktionär finns det fem kolumner ”pnr”, ”lön”, ”namn”, ”skjutbana” och ”fält”. Pnr sparas som CHAR en för att representera personnumret med ett bindestreck, personnumret är primärnyckeln. Lön sparas som INT, lönen representeras i heltal. Namn sparas i VARCHAR eftersom vi inte vet hur långt namnet är, namn representerar funktionärens namn. Skjutbana sparas som INT, heltalet representerar skjutbanans nummer. Skjutbana variabeln är en främmandenyckel, nyckeln hämtas från skjutbanas nr attribut. Fält använder en VARCHAR för att spara namnet på skjutfältet. Fält är en främmande nyckel som hämtas från skjutfälts namn attribut.

```
CREATE TABLE funktionär(
   pnr CHAR(13),
   lön INT,
```

```
    namn VARCHAR(40),
    skjutbana INT,
    fält VARCHAR(40),
    PRIMARY KEY(pnr),
    FOREIGN KEY(fält) REFERENCES skjutfält(namn),
    FOREIGN KEY(skjutbana) REFERENCES skjutbana(nr)
)ENGINE=innodb;
```

Tabellen ammunition har fyra attribut ”kaliber”, ”namn”, ”gevärNamn” och ”skyttPnr”. Kaliber sparas i en DECIMAL efterson ammunitionens kan ha storlekar används decimaler med en precision på fyra siffror totalt (från -99,99 till 99,99), varav två är decimaler. Namn sparas som VARCHAR och sparar ammunitionens namn. GevärNamn är sparade i en VARCHAR, det är en främmandenyckel som hämtar sin information från gevärs namn attribut. SkyttPnr sparas i en CHAR med 13 karaktärers begränsning, den här främmande nyckel refererar till kolumnen skyttPnr i tabellen gevär. Det existerar en indexering på namn och skyttPnr för att försnabba sökningstiden på de attributen.

Tabellen är ammunitionÄgs en många till många relations tabell, det ingår tre kolumner i den ”funkPnr”, "ammunitionNamn" och "skyttPnr". De här tre attributen skapar primärnyckeln för tabellen. Anledningen till att den här existerar är eftersom I ER-modellen finns det en många till många relation mellan funktionär och ammunition, därmed behövs det skapas en ny tabell för att kunna spegla det i en databas. FunkPnr sparas som CHAR, är en främmandenyckel och refererar till kolumnen pnr i tabellen funktionär. AmmunitionNamn sparas som VARCHAR och är en främmandenyckel som refererar till kolumnen namn i tabellen ammunition. SkyttPnr sparas i datatypen CHAR, är en främmandenyckel som refererar till attributet skyttPnr i entiteten gevär.

```
CREATE TABLE ammuntionÄgs(
        funkPnr CHAR(13),
        ammunitionNamn VARCHAR(40),
        skyttPnr CHAR(13),
        PRIMARY KEY(funkPnr, ammunitionNamn, skyttPnr),
        FOREIGN KEY(skyttPnr) REFERENCES gevär(skyttPnr),
        FOREIGN KEY(funkPnr) REFERENCES funktionär(pnr),
        FOREIGN KEY(ammunitionNamn) REFERENCES ammunition(namn)
)ENGINE=innodb;
```

Tabellen "måltavla" skapades med två kolumner, ”nr” och ”antal”. Nr av datatypen INT lagrar numret på måltavlan. Antal är av datatypen INT, lagrar antalet träffpunkter. Primärnyckeln är nr.

```
CREATE TABLE måltavla(
    nr INT,
    antal INT,
    PRIMARY KEY(nr)
```

)ENGINE=innodb;

Tabellen skjutserie är sammanställd av sex kolumner, ”starttid”, ”resultat”, ”skyttPnr”, ”bevakarPnr”, ”målNr”, ”banNr”. Primärnyckeln är attributen målNr, skyttPnr och starttid. Attributet starttid använder sig av DATETIME, detta eftersom datatypen kan spara tid och datum. Resultat sparas i en CHAR, detta är på grund av att det är svårt att veta vad som kommer vara resultatet, antingen kan resultatet vara en siffra eller så kan det vara ett ord, till exempel diskvalificerad. SkyttPnr sparas i en CHAR, det är en främmandenyckel som hämtas från tabellen skytt, kolumnen pnr. MålNr är en INT som spara vilken måltavla som skjuts på, det är främmandenyckel den kommer från kolumnen nr i tabellen måltavla. BevakarPnr är personnumret på funktionären som bevakar skjutserien, variabeln är sparad i en CHAR, det är en främmandenyckel, den kommer från funktionär tabellen och från pnr attributet. BanNr säger vilken bana skjutserien befinner sig på, variabeln är sparad i en INT, nyckeln är främmande, den hämtas från skjutbana entiteten och nr attributet.

```
CREATE TABLE skjutserie(
   starttid DATETIME,
   resultat CHAR(40),
   skyttPnr CHAR(13),
   bevakarPnr CHAR(13),
   målNr INT,
   banNr INT,
   PRIMARY KEY(målNr, skyttPnr, starttid),
   FOREIGN KEY(målNr) REFERENCES måltavla(nr),
   FOREIGN KEY(skyttPnr) REFERENCES skytt(pnr),
   FOREIGN KEY(banNr) REFERENCES skjutbana(nr),
   FOREIGN KEY(bevakarPnr) REFERENCES funktionär(pnr)
)ENGINE=innodb;
```

3. Transaktioner

Om data ska matas in till en databas används "INSERT INTO" uttrycket. Efter uttrycket kommer namnet på tabellen där vi vill införa data. Sedan, öppnas en parentes och specificeras vilka kolumner vi vill fylla sedan stängs parentesen. Därefter använder vi ”VALUES” uttrycket och matar in värden i samma ordning som har specificerats tidigare i nya parenteser där värdena ska sättas in. Detta görs för varje tabell i databasen som data ska matas in till.

Eftersom den data som var utskriven i uppgiftsbeskrivningen var i oordnad form, var det viktigt att sätta in dem i databasen på ett korrekt sätt, och på så sätt kunna undvika inkonsekvenser och bibehålla integriteten i databasen. Därmed säkerställa att främmande nycklar redan är skapade, till att de ska användas.

Först börjas data att fyllas in i skjutfält eftersom den inte har några främmande nycklar. Tabellen fylls med data enligt fråga ett, där namnet på skjutfältet ska vara ”Kråk”, ha telefonnumret ”0500-999999” och i staden ”Skövde”.

```
INSERT INTO skjutfält (namn, tele, stad)
# Information från uppgiftsbeskrivningen
VALUES ('Kråk', ’0500999999’, 'Skövde');
```

Därefter fylls det data i skytt-tabellen. Vi vet att det finns minst fem skyttar som deltar. Detta vet vi eftersom Bosse nämns som skytt i fråga tre, fråga fyra nämns en skytt med personnummer 761223-5656, i fråga sex får vi reda på att en skytt heter Allan och tillhör laget Skultorp, i fråga 8 sägs det att skytten Nisse existerar och i fråga tio sägs det att skytten Ivar bevakas.

Det lämnar en andel information utesluten. Den information som finns är att en skytt heter Bosse, med personnummer 560123-6666 och tillhör laget Göteborg. En skytt med personnummer 761223-5656. Skytten Allan med 781222-2323 och tillhör laget Skultorp. Skytten Nisse. Samt skytten Ivar. Med det sagt får informationen hittas på.

```
# Fyller tabellen skytt med data
INSERT INTO skytt(pnr, namn, skyttelag)
# Information från uppgiftsbeskrivningen
VALUES ('761223-5656', 'Emil', 'Skultorp'),
    ('560123-6666', 'Bosse', 'Göteborg'),
    ('781222-2323', 'Allan', 'Skultorp'),
    ('020910-2727', 'Nisse', 'Uppsala'),
    ('021123-8080', 'Ivar', 'Uppsala'),
# Egen data för test
    ('661211-1415', 'Mattias', 'Stockholm'),
    ('790623-7547', 'Martin', 'Göteborg'),
    ('680130-1253', 'Kerstin', 'Stockholm'),
    ('020610-3477', 'Anton', 'Uppsala'),
    ('010607-7384', 'Markus', 'Uppsala'),
    ('XXXXXX-XXXX', 'Martin', 'Rensport');
```

Sedan fortsätter data matas in efter ordning utan främmandenycklar. Det är sagt i uppgiftsbeskrivningen att måltavla existerar, i både fråga sju och nio, det står även att måltavla nummer två har fem träffpunkter. Därmed gjordes antagandet av att det finns fler än en måltavla, speciellt med tanke på att den som refereras är betecknas med nummer två.

```
# Fyller tabellen måltavla med data
INSERT INTO måltavla(nr, antal)
# Information från uppgiftsbeskrivningen
VALUES (1, 5),
```

```
    (2, 5),
    (3, 3),
    (4, 5),
    (5, 7),
    (6, 10),
# Egen data för test
    (7, 8),
    (8, 2),
    (9, 4),
    (10, 5),
    (11, 10),
    (12, 10);
```

Nu är alla utan främmandenycklar slut, därmed kan data som redan skapats fortsätta byggas på andra tabeller. I verksamheten vet skrivs det att flera skjutbanor finns. Det står också att skjutfältet Kråk har tio stycken skjutbanor. Det står också att det finns flera moment, där man antingen får stå, ligga eller sitta på knä. I fråga två står det att skjutbana ett och två, de banorna finns på Kråk. I fråga åtta står det om bana fyra.

Det innebär att vi får göra antaganden för resten av de åtta kvarstående banorna, om vart de ligger samt om vilket moment alla tio banor har. Det kan förmodas att alla tio banor ligger på Kråk, men vilket moment som de banorna har får hittas på.

```
# Fyller tabellen skjutbana med data
INSERT INTO skjutbana (nr, moment, skjutfältNamn)
# Information från uppgiftsbeskrivningen
VALUES (1, ' Knästående', 'Kråk'),
    (2, 'Sittande', 'Kråk'),
    (3, 'Stående', 'Kråk'),
    (4, 'Liggande', 'Kråk'),
    (5, 'Stående', 'Kråk'),
    (6, 'Liggande', 'Kråk'),
    (7, 'Sittande', 'Kråk'),
    (8, 'Stående', 'Kråk'),
    (9, 'Liggande', 'Kråk'),
    (10, 'Knästående', 'Kråk');
```

Lämpligt att mata in data för efter är funktionärernas, eftersom data för skjutbanan har precis skapats, vilket funktionär-tabellen använder. I uppgiftsbeskrivningen står det att det ska finnas en funktionär för varje skjutbana och i sin tur ansvara för skjutbanan. Fråga två säger att det finns två funktionärer med respektive personnummer 790129-4444 och 810912-5555, de har ansvar för skjutbana ett respektive två samt att de har 18000 i lön. Fråga sex nämns funktionären med personnummer 870923-3434. Fråga åtta nämner en funktionär med personnummer 560123-4455. Fråga tio säger att det finns en funktionär med personnummer

670809-9999.

Det här lämnar mycket data till antagande. Ingen av funktionärerna har namn. Ett fåtal av de har specificerad bana och enbart vissa funktionärer har personnummer. Två utav funktionärernas lön nämns. Det är mycket information som bygger på antagande. Det kan antas att alla funktionärer jobbar på samma skjutfält, alla har en lön, ett namn, en bana de ansvarar över mellan tre till tio och ett personnummer.

```
# Fyller tabellen funktionär med data
INSERT INTO funktionär (pnr, namn, lön, fält, skjutbana)
# Information från uppgiftsbeskrivningen
VALUES ('790129-4444', 'Erik', 18000, 'Kråk', 1),
    ('560123-4455', 'Axel', 5000, 'Kråk', 4),
    ('670809-9999', 'Filip', 5000, 'Kråk', 3),
    ('810912-5555', 'Måns', 18000, 'Kråk', 2),
    ('870923-3434', 'Per', 12000, 'Kråk', 5),
# Egen skapad data
    ('990210-7572', 'Anders', 10000, 'Kråk', 6),
    ('600708-3043', 'Jonas', 10000, 'Kråk', 7), #65000
    ('680403-8884','Axel',  4000, 'Kråk', 8),
    ('960223-3544', 'Richard', 12500, 'Kråk', 9), #30300
    ('981012-7642', 'Viktor', 16000, 'Kråk', 10), #22000
# Egen data för test
    ('690216-2379', 'Peo', 10000, 'Kråk', 6),
    ('570903-4567', 'Sam', 18000, 'Kråk', 4),
    ('651211-5633', 'Hassan', 13000, 'Kråk', 5);
```

Uppgiftsbeskrivningen nämner att alla skyttar har varsitt gevär. I fråga fyra sägs det att ett gevär med namn Izhmash, geväret väger 4.5 kilo och tillhör skytten med personnummer 761223-5656. Det sägs också att skytten Bosse använder sig av ett likadant gevär som 761223-5656. För att skapa data i gevär-tabellen behövs skytten vara skapad sedan tidigare, detta eftersom skyttPnr är en främmandenyckel.

Det här utelämnar information om resten av skytterna. Ett antagande som behövs göras då är, alla skyttar har tilldelats till ett gevär. Med tanke på att det inte nämns vilket gevär resten av skyttarna använder, hittas den datan på.

```
# Fyller tabellen gevär med data
INSERT INTO gevär (namn, vikt, skyttPnr)
# Information från uppgiftsbeskrivningen
VALUES ('Izhmash', 4.5, '560123-6666'),
    ('Izhmash', 4.5, '761223-5656'),
    ('Ruger 10/22', 6, '781222-2323'),
    ('AK74', 3.07, '020910-2727'),
```

```
    ('AK74', 3.07, '021123-8080'),
# Egen data för test
    ('Zastava M90', 4.9, '790623-7547'),
    ('Zastava M90', 4.9, '661211-1415'),
    ('G36', 3.6, '680130-1253'),
    ('Remington R4', 4.3, '020610-3477'),
    ('Remington R4', 4.3, '010607-7384');
```

Anledningen till att datan till ammunition kommer matas in efter gevär, är att det finns ett förhållande mellan de här två tabeller. Detta innebär att en viss ammunition kommer vara kopplad till ett specifikt gevär.
Det sägs att varje vapen har en tillhörande ammunition. Fråga fem nämner att Bosses gevär har, en kaliber 22, namnbeteckningen X-act och den är kopplad till Izhmash geväret. Fråga sex berättar att ammunitionen med kaliber 22 och namn Midas+ finns.

Eftersom gevären är redan skapade sedan tidigare, fylls ytterligare information på. Antaganden görs för två utav gevären, vi kan anta att Midas+ ammunitionen tillhör samma gevär. Detsamma kan göras för båda Izhmash gevären, båda bör använda sig av X-act och kaliber 22. Resten av informationen får hittas på.

```
# Fyller tabellen ammunition med data
INSERT INTO ammunition (kaliber, namn, gevärNamn, skyttPnr)
# Information från uppgiftsbeskrivningen
VALUES (22, 'X-act', 'Izhmash', '560123-6666'),
    (22, 'X-act', 'Izhmash', '761223-5656'),
    (22, 'Midas+', 'Ruger 10/22', '781222-2323'),
    (5.45, 'Hornady Black', 'AK74', '020910-2727'),
    (5.45, 'Hornady Black', 'AK74', '021123-8080'),
# Egen data för test
    (5.56, 'Prvi', 'Zastava M90', '790623-7547'),
    (5.56, 'Prvi', 'Zastava M90', '661211-1415'),
    (5.56, 'Winchester USA', 'G36', '680130-1253'),
    (5.56, 'Wolf Gold', 'Remington R4', '020610-3477'),
    (5.56, 'Wolf Gold', 'Remington R4', '010607-7384');
```

Att mata in data i ammuntionÄgs kommer efter ammunition och funktionär eftersom det är en tabell för många till många relation. Därmed bygger den på främmandenycklar från ammunition och funktionär-tabellen.
I kapitlet verksamheten sägs det, att varje funktionär ansvarar över de tävlandes ammunition. Fråga sex nämner att funktionären 870923-3434 ansvarar över Allans ammunition. Fråga åtta säger att funktionären med personnummer 560123-4455 ansvarar över Nisses ammunition.

I den kommande data in matningen behövs det göra ett antal antagande som bygger på tidigare antaganden. Den första antagandet görs baserat på infomrationen om vilken

ammunition som ägs av vilken skytt. Andra antagandet blir för de uteblivna funktionären om vilken skytts ammunition som de ansvara för.

```
# Fyller tabellen ammuntionÄgs med data
INSERT INTO ammuntionÄgs (funkPnr, skyttPnr, ammunitionNamn)
# Information från uppgiftsbeskrivningen
VALUES ('790129-4444', '560123-6666', 'X-act'),
    ('810912-5555','761223-5656', 'X-act'),
    ('870923-3434', '781222-2323','Midas+'),
    ('560123-4455', '020910-2727', 'Prvi'),
    ('670809-9999', '021123-8080', 'Prvi'),
# Egen data för test
    ('990210-7572',  '790623-7547', 'Prvi'),
    ('570903-4567', '661211-1415', 'Prvi'),
    ('651211-5633', '680130-1253', 'Winchester USA'),
    ('600708-3043', '020610-3477','Wolf Gold' ),
    ('680403-8884', '010607-7384', 'Wolf Gold');
```

Skjutserien bygger på många andra tabeller och därför kommer sist i inmatningsfasen. Datan i tabellen använder sig av specifik data, och interagerar med många olika element i databasen, det är många främmandenycklar datan beror på.
Uppgiftsbeskrivningen säger att skjutserien utförs på en skjutbana, att det finns en funktionär som ansvarar över serien, att det finns ett resultat, att det finns en skytt och att det finns en måltavla.
Fråga sju nämner att skjutserien som startade 13:01:33 med resultatet fem, som utfördes på måltavla två, den bevakades av funktionären 790129-4444 och att skytten var Bosse. Fråga åtta berättar att en skjutserie startade 12:00:34 där skytten Nisse deltog, träffade 5 punkter och funktionären 560123-4455 ansvarade och den hölls på bana fyra. Fråga tio nämner att skytten Ivar var med i en skjutserie där funktionären 670809-9999 bevakade.

Här lämnas mycket upp till den som matar in data. Det innebär att många egna antaganden får tas. Resultatet på skjutserien nämns inte för alla serier. Det antas också att varje skytt är med i en skjutserie. Datum nämns aldrigs, klockslag nämns ett fåtal gånger. Det kan antas att samma funktionär som ansvara över ammunitionen också ansvarar över skjutserien. Måltavla och bannummer nämns inte för alla.

```
INSERT INTO skjutserie (starttid, resultat, målNr, banNr, skyttPnr, bevakarPnr)
# Information från uppgiftsbeskrivningen
VALUES ('2012-03-17 13:01:33', '5', 2, 1, '560123-6666', '790129-4444'),
    ('2012-03-17 12:00:34', '5', 4, 4, '020910-2727', '560123-4455'),
    ('2012-03-10 16:05:10', '4', 3, 3, '021123-8080', '670809-9999'),
    ('2012-05-25 12:50:00', '7', 6, 5, '781222-2323', '870923-3434'),
    ('2023-05-21 15:52:10', '4', 2, 6, '761223-5656', '810912-5555'),
# Egen data för test
```

('2012-01-21 12:01:33', '5', 7, 9, '790623-7547', '990210-7572'), # 790623-7547
('2001-06-07 12:00:34', '5', 7, 10, '020610-3477', '600708-3043'),
('2002-10-25 16:05:10', '4', 9, 10, '010607-7384', '680403-8884'),
('2000-07-10 12:50:00', '7', 11, 7, '680130-1253', '651211-5633'),
('2010-09-10 15:52:10', '4', 12, 8, '661211-1415', '570903-4567');

4. Frågeoperationer

Frågeoperationer: Inkludera kod och en kort beskrivning av funktionaliteten för varje frågeoperation du har utfört.

Det kommer utföras olika frågeoperationer för att hämta, filtrera och manipulera data i databasen. Notera om någon specifik operation inte går att genomföra på grund av antigen databasdesignen eller på grund av inmatningen av data, kommer det förklaras hur det hade kunnat möjliggöras.

Fråga ett, Hämta telefonnumret till Kråks skjutfält:

```
SELECT skjutfält.tele
FROM skjutfält
WHERE namn = 'Kråk';
```

I frågeoperation används SELECT uttrycket för att hämta telefonnumret från skjutfält tabellen. Vi filtrerar sedan genom att använda WHERE uttrycket, med villkoret, där namn är lika med Kråk.

Resultat:
tele
0500999999

Fråga två, Hämta namn och lön för funktionären med personnummer 790129-4444.

```
SELECT funktionär.namn, funktionär.lön
FROM funktionär
WHERE pnr = '790129-4444';
```

Frågeoperationen använder SELECT för att välja namn och lön, sedan används FROM uttrycket för att välja varifrån attributen ska hämtas, i detta fall funktionär. Sedan filtrerar man genom WHERE, där väljer man personnumret.

Reslutat:
namn, lön
Erik, 18000

Fråga tre, Hämta funktionärens namn som ansvarar för ammunitionen "X-act".

```
SELECT funktionär.namn
FROM funktionär, ammuntionÄgs
WHERE funktionär.pnr = ammuntionÄgs.funkPnr AND ammuntionÄgs.ammunitionNamn =
'X-act';
```

Frågeoperationen använder SELECT för att hämta funktionärens namn, från tabellen funktionär. Sedan används både funktionär och ammuntionÄgs i FROM. WHERE används för att skapa en JOIN mellan funktionär och ammunition. Där vi filtrerar efter där funktionärens personnummer och ammuntionÄgs funktionärens personnumer är detsamma och där ammunitions namnet är X-act.

Resultat:
namn
Erik
Måns

Fråga fyra, Hämta personnumret för skytten som använder geväret "Izmash" och ammunitionen med kaliber 22 och namn "X-act".

```
SELECT ammunition.skyttPnr
FROM ammunition
WHERE kaliber = 22 AND namn = 'X-act' AND gevärNamn = 'Izhmash';
```

SELECT väljer skyttens personnummer, FROM väljer från ammuntion-tabellen. WHERE specificerar frågan letar efter där kaliber är lika med 22, namn är lika med X-act och där gevärets namn är Izhmash.

Resultat:
skyttPnr
560123-6666
761223-5656

Fråga fem, Hämta namn på funktionärerna som ansvarar för skjutbanan med moment "Stående"?

```
SELECT funktionär.namn
FROM funktionär, skjutbana
WHERE funktionär.skjutbana = skjutbana.nr AND skjutbana.moment = 'Stående';
```

Funktionärens namn väljs, sedan väljs det att hämta information från skjutbana och funktionär. Med WHERE uttrycket skapas en JOIN mellan funktionär och skjutbana, detta för att kunna ta reda på vilken funktionär som ansvarar över en bana med Stående moment.

Resultat:
namn
Filip
Hassan
Per
Axel

Fråga sex, Hämta personnumret för de funktionärer som har samma lön. (Tips: skapa instanser av samma tabell och jämför dess lön).

```
SELECT DISTINCT f1.pnr
FROM funktionär f1, funktionär f2
WHERE f1.lön = f2.lön
AND f1.pnr != f2.pnr;
```

Genom att använda DISTINCT med SELECT, kommer endast unika personnummer bli resultat. Det skapas två instanser av samma tabell genom att använda funktionär f1 och funktionär f2. I WHERE kollas det sedan om funktionär ett har samma lön som funktionär två, samt att de inte har samma personnummer.

Resultat:
pnr
670809-9999
810912-5555
790129-4444
560123-4455
990210-7572
570903-4567
690216-2379

Fråga sju, Hämta personnumret för de skyttar som inte har deltagit i en skjutserie.

```
SELECT skytt.pnr
FROM skytt
WHERE skytt.pnr NOT IN (SELECT skyttPnr FROM skjutserie);
```

I denna frågeoperation används en underfråga med NOT IN uttrycket. Underfrågan frågar efter alla skyttpersonnummer som deltagit i skjutserien. Genom att använda NOT IN i WHERE frågan utesluts alla personnummer som är med i underfrågan.

Resultat:
Enligt beskrivningarna i 3.2, ska det inte finnas någon som inte är med i en skjutserie. Men i test datan finns det med en skytt som inte deltar.
pnr
XXXXXX-XXXX

Fråga åtta, Hämta laget för de skyttar som träffade alla 5 måltavlor under en skjutserie.

```
SELECT skytt.skyttelag
FROM skytt
WHERE EXISTS (SELECT skyttPnr FROM skjutserie WHERE skyttPnr = skytt.pnr AND
resultat = '5');
```

Frågeoperationen hämtar skyttarnas skyttelag, sedan används WHERE uttrycket för att kolla om det finns, i en underfråga, några skyttar som har fått resultatet fem i en skjutserie. Underfrågan returneras sedan visas skyttelagen med resultat fem i skjutserien.

Resultat:
skyttelag
Göteborg
Uppsala
Uppsala
Göteborg

Fråga nio, Visa de skyttar som deltagit i en skjutserie.

```
SELECT skytt.pnr, skytt.namn
FROM skytt, skjutserie
WHERE skytt.pnr = skjutserie.skyttPnr;
```

Här hämtas namn och personnummer från skytt. I WHERE uttrycket specificeras villkoret där matchas skyttens personnummer från tabellen skytt med skyttens personnummer från skjutserie. Det här bildar en INNER JOIN och matchar personnumret på de skyttar som har deltagit i en skjutserie.

Resultat:
pnr, namn
560123-6666, Bosse
021123-8080, Ivar
020910-2727, Nisse
781222-2323, Allan
761223-5656, Emil
680130-1253, Kerstin

661211-1415, Mattias
790623-7547, Martin
020610-3477, Anton
010607-7384, Markus

Fråga tio, Visa de funktionärer som bevakat exakt två stycken skjutserier. (Tips: använd count).

```
SELECT funktionär.pnr, funktionär.namn
FROM funktionär
WHERE (
   SELECT COUNT(*)
   FROM skjutserie
   WHERE skjutserie.bevakarPnr = funktionär.pnr
) = 2;
```

I frågeoperationen används en underfråga, den räknar antalet skjutserier som varje funktionär har bevakat.
Första SELECT väljer funktionärens personnummer, sedan anges det i FROM att det är funktionär-tabellen som ska användas. I WHERE uttrycket används en underfråga. Där används COUNT som hämtar alla kolumner från skjutserie, sedan används WHERE med en JOIN på bevakarens personnummer och funktionärens personnummer.

Resultatet:
#pnr, namn

Den här är tom eftersom det inte finns varken i beskrivningen en funktionär som bevakar två skjutserier, lika så med test datan.

Fråga elva, Lista alla funktionärer och sortera namnet i omvänd ordning (Z först, A sist).

```
SELECT funktionär.namn
FROM funktionär
ORDER BY namn DESC;
```

Frågeoperationen använder ORDER BY uttrycket för att sortera resultaten i alfabetiskordning, men eftersom DESC uttrycket, uttrycket står för descending, det menas med att sortering börjar med bokstaven Z. Därmed kommer funktionärernas namn returneras med motsatt alfabetisk ordning.

Resultat:
namn
Viktor
Sam

Richard
Per
Peo
Måns
Jonas
Hassan
Filip
Erik
Axel
Axel
Anders

Fråga tolv, Hämta medelresultatet för alla skjutserier.

```
SELECT AVG(resultat)
FROM skjutserie;
```

Frågeställningen använder uttrycket AVG() för att beräkna medelvärdet av kolumnen resultat, i tabellen skjutserie.

Resultat:
AVG(resultat)
5

Fråga 13, Hämta medelresultatet för skjutserierna på respektive skjutbana. (Tips: använd Group by).

```
SELECT skjutbana.nr, AVG(skjutserie.resultat)
FROM skjutbana, skjutserie
WHERE skjutserie.banNr = skjutbana.nr
GROUP BY skjutbana.nr;
```

Frågeoperationen använder SELECT för att hämta skjutbanans nummer och medelvärdet på resultaten i skjutserien. Sedan kombineras tabellerna skjutbana och skjutserie med en JOIN. Därefter grupperas de efter resultatet för varje skjutbana.

Resultat:
nr, AVG(skjutserie.resultat)
1, 5
3, 4
4, 5
5, 7
6, 4
7, 7

8, 4
9, 5
10, 4.5

Fråga 14, Hämta all information om skyttar som tillhör ett lag som börjar med bokstaven "R". (Tips: använd funktionen rlike eller liknande).

```
SELECT *
FROM skytt
WHERE skyttelag LIKE 'R%';
```

SELECT *, hämtar alla kolumner i tabellen. FROM hämtar tabellerna från skytt. Sedan används WHERE uttrycket på skyttelag, där används LIKE uttrycket för hitta ett specificerad mönster, i detta fall, skyttelag som börjar på ”R”, procent tecknet specificerar att matcha alla tecken som kommer efter r:et.

Resultat:
pnr, namn, skyttelag
XXXXXX-XXXX, Martin, Rensport

Fråga 15, Hämta namn och lag för de skyttar som angivit ett personnummer som inte är på formen "XXXXXX-XXXX" där X är en siffra mellan 0 och 9. (Tips: använd rlike eller liknande).

```
SELECT skytt.namn, skytt.skyttelag
FROM skytt
WHERE pnr NOT RLIKE '^[0-9]{6}-[0-9]{4}$';
```

Först används SELECT för att välja skyttens namn och lag. De hämtas från tabellen skytt. WHERE uttrycket letar efter personnummer (pnr) som inte (NOT uttrycket) matchar uttrycket '^[0-9]{6}-[0-9]{4}$' matchar mönster med sex siffror följt av bindestreck och sedan fyra siffror.

Resultat:
namn, skyttelag
Martin, Rensport

Fråga 16, Hämta namnet på den funktionär som tjänar mest med hjälp av kommandot MAX.

```
SELECT funktionär.namn
FROM funktionär
WHERE lön = (
        SELECT MAX(lön) FROM funktionär
);
```

I frågeoperationen säger SELECT att välja funktionärens namn. FROM specificerat att välja från funktionär-tabellen. WHERE letar i underfrågan där lönen är högst, SELECT MAX(lön) FROM funktionär returnerar det högsta värdet från lön attributet i funktionär-tabellen.
Resultat:
namn
Jonas

Fråga 17, Hämta namnet på den skytt som senast startade en skjutserie.

```
SELECT skytt.namn
FROM skytt, skjutserie
WHERE skytt.pnr = skjutserie.skyttPnr
ORDER BY skjutserie.starttid DESC
LIMIT 1;
```

Frågeoperationen börjar med att välja skyttens namn (SELECT skytt.namn), sedan anges tabellerna som data ska hämtas ifrån, (FROM skyttm skjutserie), WHERE väljer att ska en JOIN mellan skjutseriens skyttPnr och skyttens personnummer (WHERE skytt.pnr = skjutserie.skyttPnr). Därefter sorteras skjutserien efter starttid, den som börjar senast kommer först (ORDER BY skjutserie.starttid DESC). Sedan begränsas det till en rad genom att använda LIMIT uttrycket.

Resultat:
namn
Emil

Fråga 18, Lista de skjutserier som påbörjat den senaste dagen. (Tips: använd curdate() eller liknande).

```
SELECT *
FROM skjutserie
WHERE DATE(starttid) = CURDATE();
```

SELECT * väljer alla kolumner. Kolumner specificeras att komma från skjutserie (FROM skjutserie). Kombinationen av WHERE uttrycket och DATE med inmatningen av starttid kommer startdatumet för skjutserien enbart visa datumet, inte tiden. Om den matchar dagens datum, vilket tas reda på genom CURDATE, då kommer det returneras de skjutserier som spelades på dagens datum.

Resultat:
starttid, resultat, skyttPnr, bevakarPnr, målNr, banNr
2023-05-21 15:52:10, 4, 761223-5656, 810912-5555, 2, 6

Fråga 19, Öka lönen för funktionärerna med en lön mellan 10000 och 12000kr med 3%.

```
CREATE TEMPORARY TABLE temp_funktionär AS
SELECT pnr, lön * 1.03 AS new_lön
FROM funktionär
WHERE lön BETWEEN 10000 AND 12000;

UPDATE funktionär, temp_funktionär
SET funktionär.lön = temp_funktionär.new_lön
WHERE funktionär.pnr = temp_funktionär.pnr;

DROP TEMPORARY TABLE temp_funktionär;
```

För att uppdatera ett värde skapas en temporär tabell, detta undviker eventuella korupptioner av data. En tillfällig tabell vid namn temp_funktionär som innehåller personnumret och den nya lönen för alla som har en lön mellan 10000 och 12000. Sedan uppdateras lönen för alla funktionärer i funktionär-tabellen, vars personnummer finns i temp_funktionär-tabellen, med de nya lönerna. Till sist tas den temporära tabellen bort eftersom den har gjort sitt jobb.

Fråga 20, Ta bort måltavlan med nummer 8.

```
DELETE FROM måltavla
WHERE nr = 8;
```

Den här frågeställningen tar bort måltavla åtta. Detta genom DELETE uttrycket, sedan specificerat varifrån (FROM måltavla). Sedan specificeras vilket nummer som ska tas bort (WHERE nr = 8;).

Den här frågeoperationen fungerar endast om det existerar en måltavla med nummer 8.

Fråga 21, Ta bort skjutserien med starttid "2012-01-21 12:01:33" som tillhör skytten med personnummer "560123-6666". (Om det inte fungerar, beskriv varför).

```
DELETE FROM skjutserie
WHERE starttid = '2012-01-21 12:01:33' AND skyttPnr = '560123-6666';
```

DELETE FROM skjutserie säger att ta bort data från skjutserie-tabellen. WHERE starttid[…] specificerar vilka kolumner som ska tas bort.

Det finns ingen post som matchar de specificerade villkoren. Det kan också bero på om man använder DATETIME i sin databastabell när den skapas, eller att det inte existerar något datum enbart tid.

5. Grundläggande förståelse

Datatyperna CHAR och VARCHAR är liknande, det finns vissa saker som skiljer dem åt. När en CHAR definieras måste det anges en maximal längd på strängen. Om en sträng inmatad skulle vara kortare än den fördefinierade maximala längden, kommer strängen fyllas ut med mellanslag. Medan en VARCHAR lagrar också strängar. Det sätts också en maximal längd på datatypen. Skillnaden är när inmatningen är kortare än maximala längden, då kommer inga mellanslag fylla ut resten av utrymmet. Det här sparar på lagringsutrymme.

Om en värdet skulle sättas för högt kommer strängen fyllas ut med onödiga mellanslag. Det här leder till en ökad storlek i minnet, det här är onödigt eftersom tabellerna tar upp mera utrymme än nödvändigt. Det kan också orsaka problem vid datavalidering och jämförelser, eftersom de extra mellanslagen räknas med, vilket kan ge felaktiga resultat.

NULL-värden kan vara problematiska. Det är på grund av att de kan vara svåra att tyda. Ett NULL-värde kan betyda att värdet är okänt, värdet är inte inmatat ännu eller att värdet är inmatat men det är inte relevant. Om en frågeställning utförs på ett attribut med NULL-värden, kan man få tillbaka ett felaktigt resultat. NULL-värden tar också upp onödig plats i minnet.
NULL-värden kan undvikas i en relationsdatamodell genom att designa en separat tabell för värden som möjligtvis inkludera NULL-värden, i tabellen finns primärnyckeln med i.

6. XML

HTML är inte designat för att strukturera data från en databas. Det är däremot skapat för att skapa struktur och form för hemsidor. HTML har fördefinierade taggar, de går inte att ändra på. Det här gör HTML svårt att anpassa till specifika datastrukturer. Tillskillnad är XML skapat för att lagra och transportera data. Det är lätt att manipulera datan i XML-filer. XML används ofta för datautbyte, konfigurationsfiler, databashantering och dokumenthantering.

Trasiga XML-filer kan bero på ett antal olika saker. Det kan uppstå på grund av mänskliga fel, stavfel eller en felaktig strukturering i taggen. Systemfel kan orsaka att start och sluttaggarna inte matchar. Kan också uppstå från fel vid dataöverföring, eller inkompatibilitet mellan de olika systemen.
Det här kan förhindras genom att ha som tumregel att alltid validera via en DTD. DTD definierar den korrekta strukturen och syntaxen för XML-dokumentet. DTD kan hjälpa till med att upptäcka och ändra eventuella fel innan XML-dokumentet används.

Att konstruera en XML-fil från ett utgivet DTD är simpelt. Eftersom DTD fungerar som ett schema för hur XML-filen ska set ut. ELEMENT-taggen i DTD berättar vad XML-elementet ska innehålla, samt ATTLIST-taggen berättar vilka attribut som elementet ska innehålla.

XML-filen börjas med att skapa rotelementet skjutfält, den har attributet namn enligt ATTLIST:en.
Därefter skapas funktionär-elementet, med det unika attribut personnummer, det skapas också två underelement ett namn och lön.
Skjutbana-elementet följer samma process, det fick ett unikt nummerattribut. Med ett under element, moment.

```
<!DOCTYPE skjutfalt SYSTEM "skjutfalt.dtd">
<skjutfalt namn="Kråk">
  <funktionar pnr="810912-5555">
    <namn>Måns</namn>
    <lon>18000</lon>
  </funktionar>
  <funktionar pnr="670809-9999">
    <namn>Filip</namn>
    <lon>5000</lon>
  </funktionar>
  <funktionar pnr="790129-4444">
    <namn>Erik</namn>
    <lon>18000</lon>
  </funktionar>
  <skjutbana nr="2">
    <moment>Sittande</moment>
  </skjutbana>
  <skjutbana nr="3">
    <moment>Stående</moment>
  </skjutbana>¨
  <skjutbana nr="1">
    <moment>Knästående</moment>
  </skjutbana>
</skjutfalt>
```

Det skulle läggas till skjutserie, måltavla och skytt i DTD-modellen. Skjutserie elementet är utformat så det inkluderas information om skytten, resultatet och måltavlan som användes, det här är då tre underelement. Skytt blev i sin tur ett element, det blev tre underelement, personnummer, namn och lag. De tre har varsitt element. Måltavla element kommer ha noll eller flera träffpunkt-element. Träffpunkt elementet har ett nummerattribut för att visa unika träffpunkter. Träffpunkt är EMPTY eftersom den inte kommer ha några underelement.

```
<!ELEMENT personnummer (#PCDATA)>
<!ELEMENT namn (#PCDATA)>
<!ELEMENT lag (#PCDATA)>
<!ELEMENT skytt (personnummer, namn, lag)>
<!ELEMENT resultat (#PCDATA)>
```

```
<!ELEMENT maltavla (traffpunkt*)>
<!ELEMENT traffpunkt EMPTY>
<!ATTLIST traffpunkt nummer CDATA #REQUIRED>
<!ELEMENT skjutserie (skytt, resultat, maltavla)>
```

BILAGOR

```
# Radera databasen om den redan existerar
DROP DATABASE IF EXISTS a22hanfa;

# Skapa en ny databas
CREATE DATABASE a22hanfa;

# Använd den nyskapade databasen
USE a22hanfa;

# Skapa tabellen skjutfält
CREATE TABLE skjutfält(
   namn VARCHAR(40),
   tele VARCHAR(40),
   stad VARCHAR(40),
   PRIMARY KEY(namn)
)ENGINE=innodb;

# Skapa tabellen skjutbana
CREATE TABLE skjutbana(
   nr INT,
   moment VARCHAR(40),
   skjutfältNamn VARCHAR(40),
   PRIMARY KEY(nr),
   FOREIGN KEY(skjutfältNamn) REFERENCES skjutfält(namn)
)ENGINE=innodb;

# Skapa tabellen skytt
CREATE TABLE skytt(
   pnr CHAR(13),
   namn VARCHAR(40),
   skyttelag VARCHAR(40),
   PRIMARY KEY(pnr),
   INDEX (pnr)
)ENGINE=innodb;

# Skapa tabellen gevär
```

```
CREATE TABLE gevär(
   namn VARCHAR(40),
   vikt DECIMAL(5,2),
   skyttPnr CHAR(13),
   PRIMARY KEY(namn, skyttPnr),
   FOREIGN KEY(skyttPnr) REFERENCES skytt(pnr),
   INDEX (namn, skyttPnr)
)ENGINE=innodb;

# Skapa tabellen funktionär
CREATE TABLE funktionär(
   pnr CHAR(13),
   lön INT,
   namn VARCHAR(40),
   skjutbana INT,
   fält VARCHAR(40),
   PRIMARY KEY(pnr),
   FOREIGN KEY(fält) REFERENCES skjutfält(namn),
   FOREIGN KEY(skjutbana) REFERENCES skjutbana(nr)
)ENGINE=innodb;

# Skapa tabellen ammunition
CREATE TABLE ammunition(
   kaliber DECIMAL(4,2),
   namn VARCHAR(40),
        gevärNamn VARCHAR(40),
   skyttPnr CHAR(13),
   INDEX (namn, skyttPnr),
   PRIMARY KEY(kaliber, gevärNamn, skyttPnr),
        FOREIGN KEY(gevärNamn, skyttPnr) REFERENCES gevär(namn, skyttPnr)
)ENGINE=innodb;

# Skapa tabellen ammuntionÄgs, många-många relation tabell
CREATE TABLE ammuntionÄgs(
        funkPnr CHAR(13),
        ammunitionNamn VARCHAR(40),
   skyttPnr CHAR(13),
   PRIMARY KEY(funkPnr, ammunitionNamn, skyttPnr),
        FOREIGN KEY(skyttPnr) REFERENCES gevär(skyttPnr),
   FOREIGN KEY(funkPnr) REFERENCES funktionär(pnr),
        FOREIGN KEY(ammunitionNamn) REFERENCES ammunition(namn)
)ENGINE=innodb;

# Skapa tabellen måltavla
```

```
CREATE TABLE måltavla(
   nr INT,
   antal INT,
   PRIMARY KEY(nr)
)ENGINE=innodb;

# Skapa tabellen skjutserie
CREATE TABLE skjutserie(
   starttid DATETIME,
   resultat CHAR(40),
   skyttPnr CHAR(13),
   bevakarPnr CHAR(13),
   målNr INT,
   banNr INT,
   PRIMARY KEY(målNr, skyttPnr, starttid),
   FOREIGN KEY(målNr) REFERENCES måltavla(nr),
   FOREIGN KEY(skyttPnr) REFERENCES skytt(pnr),
   FOREIGN KEY(banNr) REFERENCES skjutbana(nr),
   FOREIGN KEY(bevakarPnr) REFERENCES funktionär(pnr)
)ENGINE=innodb;

# Fyller tabellen skjutfält med data
INSERT INTO skjutfält (namn, tele, stad)
# Information från uppgiftsbeskrivningen
VALUES ('Kråk', '0500999999', 'Skövde');

# Fyller tabellen skytt med data
INSERT INTO skytt(pnr, namn, skyttelag)
# Information från uppgiftsbeskrivningen
VALUES ('761223-5656', 'Emil', 'Skultorp'),
      ('560123-6666', 'Bosse', 'Göteborg'),
      ('781222-2323', 'Allan', 'Skultorp'),
      ('020910-2727', 'Nisse', 'Uppsala'),
      ('021123-8080', 'Ivar', 'Uppsala'),
# Egen data för test
      ('661211-1415', 'Mattias', 'Stockholm'),
      ('790623-7547', 'Martin', 'Göteborg'),
      ('680130-1253', 'Kerstin', 'Stockholm'),
      ('020610-3477', 'Anton', 'Uppsala'),
      ('010607-7384', 'Markus', 'Uppsala'),
      ('XXXXXX-XXXX', 'Martin', 'Rensport');

# Fyller tabellen måltavla med data
INSERT INTO måltavla(nr, antal)
```

```
# Information från uppgiftsbeskrivningen
VALUES (1, 5),
      (2, 5),
      (3, 3),
      (4, 5),
      (5, 7),
      (6, 10),
# Egen data för test
      (7, 8),
      (8, 2),
      (9, 4),
      (10, 5),
      (11, 10),
      (12, 10);

# Fyller tabellen skjutbana med data
INSERT INTO skjutbana (nr, moment, skjutfältNamn)
# Information från uppgiftsbeskrivningen
VALUES (1, 'Knästående', 'Kråk'),
      (2, 'Sittande', 'Kråk'),
      (3, 'Stående', 'Kråk'),
      (4, 'Liggande', 'Kråk'),
      (5, 'Stående', 'Kråk'),
      (6, 'Liggande', 'Kråk'),
      (7, 'Sittande', 'Kråk'),
      (8, 'Stående', 'Kråk'),
      (9, 'Liggande', 'Kråk'),
      (10, 'Knästående', 'Kråk');

# Fyller tabellen funktionär med data
INSERT INTO funktionär (pnr, namn, lön, fält, skjutbana)
# Information från uppgiftsbeskrivningen
VALUES ('790129-4444', 'Erik', 18000, 'Kråk', 1),
      ('560123-4455', 'Axel', 5000, 'Kråk', 4),
      ('670809-9999', 'Filip', 5000, 'Kråk', 3),
      ('810912-5555', 'Måns', 18000, 'Kråk', 2),
      ('870923-3434', 'Per', 12000, 'Kråk', 5),
# Egen skapad data
      ('990210-7572', 'Anders', 10000, 'Kråk', 6),
      ('600708-3043', 'Jonas', 10000, 'Kråk', 7), #65000
      ('680403-8884','Axel',  4000, 'Kråk', 8),
      ('960223-3544', 'Richard', 12500, 'Kråk', 9), #30300
      ('981012-7642', 'Viktor', 16000, 'Kråk', 10), #22000
# Egen data för test
```

```
      ('690216-2379', 'Peo', 10000, 'Kråk', 6),
      ('570903-4567', 'Sam', 18000, 'Kråk', 4),
      ('651211-5633', 'Hassan', 13000, 'Kråk', 5);

# Fyller tabellen gevär med data
INSERT INTO gevär (namn, vikt, skyttPnr)
# Information från uppgiftsbeskrivningen
VALUES ('Izhmash', 4.5, '560123-6666'),
      ('Izhmash', 4.5, '761223-5656'),
      ('Ruger 10/22', 6, '781222-2323'),
      ('AK74', 3.07, '020910-2727'),
      ('AK74', 3.07, '021123-8080'),
# Egen data för test
            ('Zastava M90', 4.9, '790623-7547'),
      ('Zastava M90', 4.9, '661211-1415'),
      ('G36', 3.6, '680130-1253'),
      ('Remington R4', 4.3, '020610-3477'),
      ('Remington R4', 4.3, '010607-7384');

# Fyller tabellen ammunition med data
INSERT INTO ammunition (kaliber, namn, gevärNamn, skyttPnr)
# Information från uppgiftsbeskrivningen
VALUES (22, 'X-act', 'Izhmash', '560123-6666'),
      (22, 'X-act', 'Izhmash', '761223-5656'),
      (22, 'Midas+', 'Ruger 10/22', '781222-2323'),
      (5.45, 'Hornady Black', 'AK74', '020910-2727'),
      (5.45, 'Hornady Black', 'AK74', '021123-8080'),
# Egen data för test
      (5.56, 'Prvi', 'Zastava M90', '790623-7547'),
      (5.56, 'Prvi', 'Zastava M90', '661211-1415'),
      (5.56, 'Winchester USA', 'G36', '680130-1253'),
      (5.56, 'Wolf Gold', 'Remington R4', '020610-3477'),
      (5.56, 'Wolf Gold', 'Remington R4', '010607-7384');

# Fyller tabellen ammuntionÄgs med data
INSERT INTO ammuntionÄgs (funkPnr, skyttPnr, ammunitionNamn)
# Information från uppgiftsbeskrivningen
VALUES ('790129-4444', '560123-6666', 'X-act'),
      ('810912-5555','761223-5656', 'X-act'),
      ('870923-3434', '781222-2323','Midas+'),
      ('560123-4455', '020910-2727', 'Prvi'),
      ('670809-9999', '021123-8080', 'Prvi'),
# Egen data för test
      ('990210-7572',  '790623-7547', 'Prvi'),
```

```
    ('570903-4567', '661211-1415', 'Prvi'),
    ('651211-5633', '680130-1253', 'Winchester USA'),
    ('600708-3043', '020610-3477','Wolf Gold' ),
    ('680403-8884', '010607-7384', 'Wolf Gold');

# Fyller tabellen skjutserie med data
INSERT INTO skjutserie (starttid, resultat, målNr, banNr, skyttPnr, bevakarPnr)
# Information från uppgiftsbeskrivningen
VALUES ('2012-03-17 13:01:33', '5', 2, 1, '560123-6666', '790129-4444'),
    ('2012-03-17 12:00:34', '5', 4, 4, '020910-2727', '560123-4455'),
    ('2012-03-10 16:05:10', '4', 3, 3, '021123-8080', '670809-9999'),
    ('2012-05-25 12:50:00', '7', 6, 5, '781222-2323', '870923-3434'),
    ('2023-05-21 15:52:10', '4', 2, 6, '761223-5656', '810912-5555'),
# Egen data för test
    ('2012-01-21 12:01:33', '5', 7, 9, '790623-7547', '990210-7572'), # 790623-7547
    ('2001-06-07 12:00:34', '5', 7, 10, '020610-3477', '600708-3043'),
    ('2002-10-25 16:05:10', '4', 9, 10, '010607-7384', '680403-8884'),
    ('2000-07-10 12:50:00', '7', 11, 7, '680130-1253', '651211-5633'),
    ('2010-09-10 15:52:10', '4', 12, 8, '661211-1415', '570903-4567');

#1.
SELECT skjutfält.tele
FROM skjutfält
WHERE namn = 'Kråk';

#2.
SELECT funktionär.namn, funktionär.lön
FROM funktionär
WHERE pnr = '790129-4444';

#3.
SELECT funktionär.namn
FROM funktionär, ammuntionÄgs
WHERE funktionär.pnr = ammuntionÄgs.funkPnr AND ammuntionÄgs.ammunitionNamn =
'X-act';

#4.
SELECT ammunition.skyttPnr
FROM ammunition
WHERE kaliber = 22 AND namn = 'X-act' AND gevärNamn = 'Izhmash';

#5.
SELECT funktionär.namn
FROM funktionär, skjutbana
```

```
WHERE funktionär.skjutbana = skjutbana.nr AND skjutbana.moment = 'Stående';

#6.
SELECT DISTINCT f1.pnr
FROM funktionär f1, funktionär f2
WHERE f1.lön = f2.lön
AND f1.pnr != f2.pnr;

#7.
SELECT skytt.pnr
FROM skytt
WHERE skytt.pnr NOT IN (SELECT skyttPnr FROM skjutserie);


#8.
SELECT skytt.skyttelag
FROM skytt
WHERE EXISTS (SELECT skyttPnr FROM skjutserie WHERE skyttPnr = skytt.pnr AND
resultat = '5');

#9.
SELECT skytt.pnr, skytt.namn
FROM skytt, skjutserie
WHERE skytt.pnr = skjutserie.skyttPnr;

#10.
SELECT funktionär.pnr, funktionär.namn
FROM funktionär
WHERE (
   SELECT COUNT(*)
   FROM skjutserie
   WHERE skjutserie.bevakarPnr = funktionär.pnr
) = 2;

#11.
SELECT funktionär.namn
FROM funktionär
ORDER BY namn DESC;

#12.
SELECT AVG(resultat)
FROM skjutserie;

#13.
```

```
SELECT skjutbana.nr, AVG(skjutserie.resultat)
FROM skjutbana, skjutserie
WHERE skjutserie.banNr = skjutbana.nr
GROUP BY skjutbana.nr;


#14.
SELECT *
FROM skytt
WHERE skyttelag LIKE 'R%';

#15.
SELECT skytt.namn, skytt.skyttelag
FROM skytt
WHERE pnr NOT RLIKE '^[0-9]{6}-[0-9]{4}$';

#16.
SELECT funktionär.namn
FROM funktionär
WHERE lön = (
        SELECT MAX(lön) FROM funktionär
);

#17.
SELECT skytt.namn
FROM skytt, skjutserie
WHERE skytt.pnr = skjutserie.skyttPnr
ORDER BY skjutserie.starttid DESC
LIMIT 1;

#18.
SELECT *
FROM skjutserie
WHERE DATE(starttid) = CURDATE();

#19.
/*UPDATE funktionär
SET lön = lön * 1.03
WHERE lön BETWEEN 10000 AND 12000;*/

CREATE TEMPORARY TABLE temp_funktionär AS
SELECT pnr, lön * 1.03 AS new_lön
FROM funktionär
WHERE lön BETWEEN 10000 AND 12000;
```

```
UPDATE funktionär, temp_funktionär
SET funktionär.lön = temp_funktionär.new_lön
WHERE funktionär.pnr = temp_funktionär.pnr;

DROP TEMPORARY TABLE temp_funktionär;
```

#20.

```
DELETE FROM måltavla
WHERE nr = 8;
```

#21.

```
DELETE FROM skjutserie
WHERE starttid = '2012-01-21 12:01:33' AND skyttPnr = '560123-6666';
```