Databassystem

# Lab assignment: SQL

Emma Jonsson

Webbutveckling – Webbplatsdesign
VT 2012

# Innehåll

# 1 Inledning

Denna rapport sammanfattar resultatet av en uppgift i kursen Databassystem som gick ut på att realisera en databas utifrån en given verksamhetsbeskrivning och mer exakt att skapa en databas åt en fiktiv filmuthyrningskedja. Målet var att få erfarenhet i databasskapande samt i hur man formulerar transaktioner och uppdateringsoperationer.

# 2 Modellering av domän

Det första steget var att modellera den domän som sedan skulle utgöra grunden till databasen. Detta skulle göras i form av en ER-modell som sedan skulle överföras till en relationsdatamodell. Det objekt som skulle finnas med och de villkor som skulle uppfyllas listas i Bilaga A.

## 2.1 ER-modellering

Som start skapades entiteten ”videobutik”. Den fick attributen namn, tel och adress där namn blev understruket för att markera att detta var entitetens nyckel, dvs det som identifierade entitetet. Därefter skapades relationen ”jobbar” som dels kopplades samman med dubbla streck till videobutiken för att markera att videobutiken **kan** ha någon som jobbar där, och dels kopplades samman med ett enkelstreck till en ny entitet som fick heta anställd. Detta markerade alltså att en anställd **kunde** jobba på en videobutik.
Videobutiken fick även en enkelstreckad relation till relationen ”har” (dvs. har hylla) och till relationen ”har” (dvs. har filmEx) för att markera att den **kunde** ha en hylla och **kunde** ha ett filmex.

Därefter färdigställdes entiteten ”anställd”. Den fick attributet a-nr understruket eftersom den identifieras med det och fick sedan även attributen namn och lön. Dessutom fick den en rekursiv relation som döptes till ”chef” för att markera att en anställd både kunde vara och ha en chef som ju faktiskt var en annan anställd. Till sist fick den en enkelstreckad relation till ”ansvarar” (dvs. ansvarar för kampanj) för att markera att en anställd **kunde** vara ansvarig för en kampanj.

Därefter färdigställdes entiteten ”kampanj” som fick attributen namn och vecko-nr där namn blev understruket. Den fick sedan en enkel relation till relationen ”ansvarar” (dvs. ansvarar för kampanj) för att markera att den **kunde** ha en anställd som var ansvarig för kampanjen. Till sist fick den en enkel relation till ”ingå” (dvs. film som ingår i kampanj) eftersom en kampanj **kunde** innefatta en film.

Därefter färdigställdes entiteten ”film”. Den fick attributet titel understruket, eftersom den identifieras utav det, och fick även attributen längd och kategori. Sedan fick den en enkel relation till ”ingå” för att markera att den **kunde** ingå i en kampanj och dubbelstreckad relation till relationen ”ge ut” (dvs. utgivare som ger ut en film) för att markera att den **måste** ha en utgivare. Till sist fick den enkel relation till ”är” (dvs. filmex är följande film) för att markera att en film **kunde** släppts som ett filmex.

Därefter färdigställdes entiteten ”utgivare”. Den fick attributen namn och tel där namn blev understruket och fick sedan en enkel relation till ”ge ut” för att markera att utgivaren **kunde** ge ut en film.

Därefter färdigställdes entiteten ”film-ex”. Eftersom den identifieras av sitt ex-nr men även kräver ytterligare identifiering för att bli unik, då den dels kan ha olika exemplarnummer i olika butiker och dels kan bestå av olika titlar men ha samma exemplarnummer och finnas i samma butik, så är den en svag entitet vilket dels markerades med att den omgavs av dubbla boxar, dels att nyckelattributet som är en del av det som identifierar den, markerades med en streckad understrykning. Attributet ex-nr blev alltså streckat understruket och dessutom fick film-ex attributen typ och pris. Därefter fick den en enkel relation till ”består av”

(dvs. uthyrning består av följande film) och en enkel relation till ”finns” (dvs. film finns i hylla) för att markera att ett filmex **kan** ingå i en uthyrning och ett filmex **kan** stå i en hylla. Dessutom fick den en dubbelstreckad relation till ”är” (dvs. film-ex är följande film) för att markera att filmexet **måste** vara en film och sedan fick även relationen ”är” ytterligare en diamant runt sig för att markera att denne hjälper en entitet att identifieras, i detta fall använde ju filmex sig utav filmens identifikation (primärnyckel) för att identifieras. Samma sak gjordes mellan filmex och videobutiken eftersom även videobutikens identifikation (primärnyckel) krävs för att identifiera ett unikt filmex och därmed visar detta även att ett filmex **måste** ägas av en videobutik.

Därefter färdigställdes entiteten ”hylla”. Även denna entitet är svag och därför måste den omges av dubbla boxar. Detta för att en hylla är unik först när man vet vilken butik den står i. Den identifieras alltså av sitt hyll-nr vilket markerades med streckad understrykning men även utav vilken butik den står i vilket markerades genom en dubbelstreckad relation till ”har” (dvs. butik har hylla) eftersom den **måste** ha en ägarbutik för att identifieras. Eftersom relationen ”har” hjälper hyllan att bli identifierad så omgavs den av dubbla diamanter. Den fick även en enkel relation till ”finns” (dvs. film finns i hylla) eftersom den **kan** innehålla ett filmex.

Därefter färdigställdes entiteten ”uthyrning”. Eftersom en uthyrning blir unik först när den innefattar en unik kund och ett unikt filmex så betyder det att den är en svag entitet som alltså måste omges av dubbla boxar. Attributet ut-datum som uthyrningen partiellt identifieras av markerades med en streckad understrykning och sedan fick den även attributet åter-datum som markerar när filmen senast skall vara tillbaka och attributet ”faktiskt tillbaka” som markerar när filmen faktiskt blev återlämnad. Därefter fick den en dubbelstreckad relation till ”består av” (dvs. uthyrning innefattar film) eftersom den **måste** innefatta en film och relationen ”består av” markerades med dubbla diamanter för att markera att den hjälper uthyrningen att bli unik och identifieras med hjälp av filmexet. Sedan fick den en likadan relation till kund eftersom den **måste** tillhöra en kund och därmed identifieras delvis utav detta. Till sist fick den en dubbelstreckad relation till ”indriver” (dvs. inkassofirma indriver uthyrning) för att markera att den **måste** sammankopplas till en inkassofirma som i det fallet att filmen blir försent inlämnad kan indriva straffavgiften.

Därefter färdigställdes entiteten ”kund” som fick attributen p-nr, namn, tel och omdöme där p-nr markerades med en understrykning då detta används för att identifiera kunden. Den fick även en enkel relation till ”hyr” eftersom en kund **kan** hyra en film.

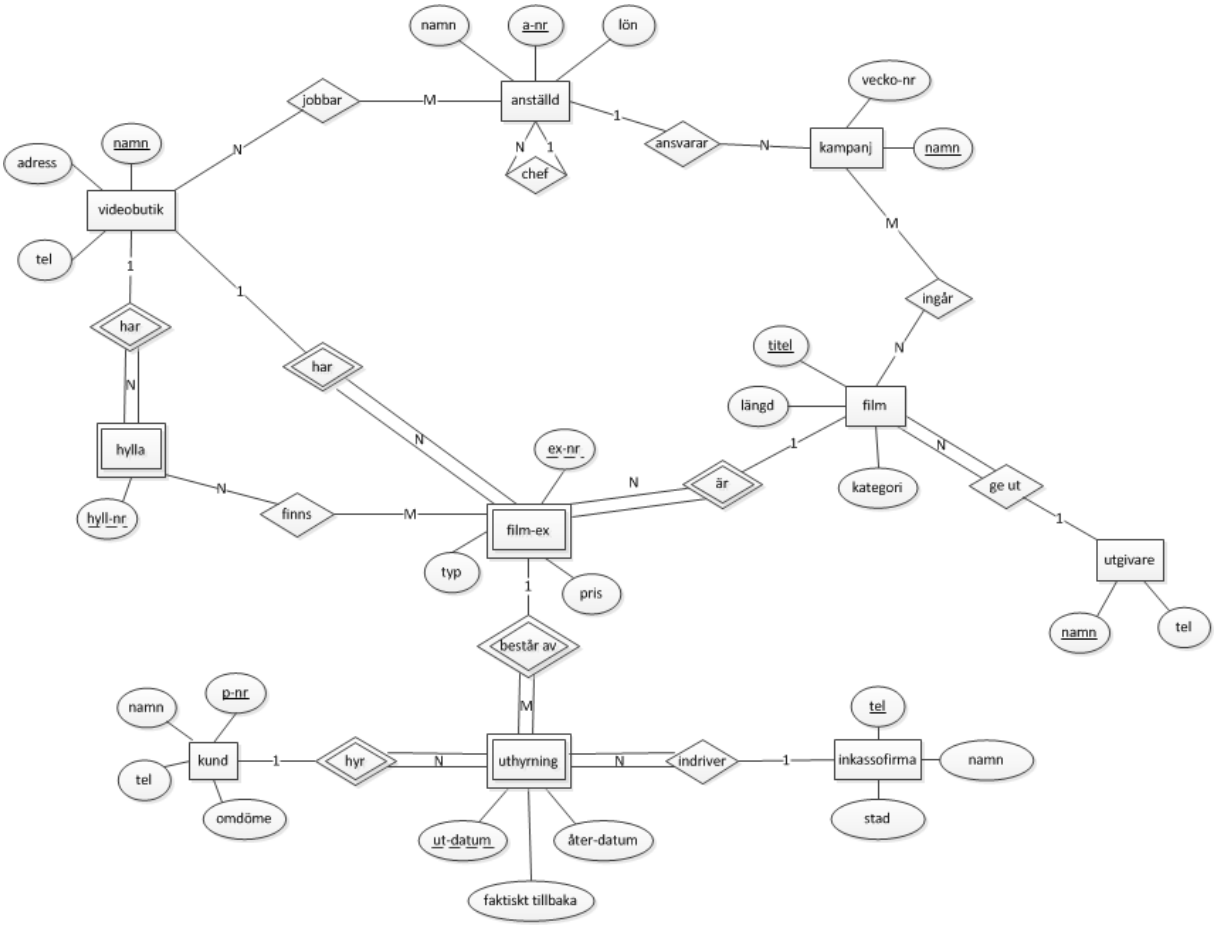
Därefter färdigställdes entiteten inkassofirma som fick attributen tel, namn och stad där tel blev understruken eftersom det är detta som identifierar inkassofirman. Den fick även en enkel relation till ”indriver” eftersom en inkassofirma **kan** vara sammankopplad till en eventuell indrivning av en uthyrning.

Till sist sattes kardinaliteten ut enligt följande:

- En videobutik kan ha många anställda, och en anställd kan jobba på många olika butiker.
- En anställd kan ha en chef, och en chef kan chefa över många anställda.
- En anställd kan ansvara över många kampanjer, och en kampanj kan ansvaras för av en anställd.
- En kampanj kan innefatta många filmer, och en film kan ingå i många kampanjer.
- En film måste ges ut av en utgivare, och en utgivare kan ge ut många filmer.
- Ett filmexemplar kan ha givits ut i många filmexemplar, och ett filmexemplar måste ha givits ut av en utgivare.
- Ett filmexemplar måste ägas utav en butik, och en butik kan äga många filmexemplar.
- Ett filmexemplar kan stå i många hyllor, och en hylla kan innefatta många filmexemplar.
- En hylla måste ägas av en videobutik, och en videobutik kan äga många hyllor.

- En uthyrning måste innefatta ett filmexemplar, och ett filmexemplar kan ingå i många uthyrningar.
- En kund kan hyra flera filmer (uthyrningar), och en uthyrning måste ha en kund som hyrt.
- En inkassofirma kan ansvara för många indrivningar kopplade till olika uthyrningar, och en uthyrning måste ha en inkassofirma kopplad till sig.

Resultatet blev:




## 2.2 Relationsdatamodellering

När det var dags omvandla ER-modellen till en relationsdatamodell gjordes det enligt följande:

Videobutiken, utgivaren, kunden och inkassofirman var alla starka och enkla entiteter som inte behövde något annat för att identifieras. Därmed skrevs först namnet på entiteten ut, följt av en parentes och inuti parentesen skrevs sedan entitetens attribut ut samt att dess nyckelattribut ströks under.

Anställd, kampanj och film såg lite annorlunda ut. Alla dessa hade något ytterligare sammankopplat till sig. En så kallad ”foreign key”. Om den anställde hade en chef så behövde ju denne identifieras i den unika raden och därav plockades ordet chef in som en nyckel. Att det var ett attribut som inte direkt hörde till entiteten (foreign key) markerades med att attributet fick ett streck ovanför sig. Det blev dock inte understruket eftersom det ju inte krävdes för att identifiera den unike anställde.

Samma sak gällde för kampanj. Om kampanjen hade en anställd som var ansvarig så plockades den anställdes unika identifikation in (i detta fall anställningsnumret) som en foreign key. Detta för att kunna markera vilken unik anställd som var ansvarig för kampanjen.
Även för film krävdes det att man plockade in en foreign key. Filmen var ju tvungen att ha en utgivare så utgivarens titel (vilket identifierar utgivaren) plockades här in som foreign key.

Eftersom jobbar, ingår och finns alla är många-till-många-relationer så blev dessa egna relationsdatamodeller trots att de egentligen inte är några egna entiteter.
Jobbar plockade in identifikationen från videobutik och anställd, dvs. namn och a-nr, som foreign keys. Dessa blev även primärnycklar för att kunna användas för att identifiera vilken unik anställd som jobbade på vilken unik videobutik. Bägge attributen markerades alltså som foreign keys med streck ovanför sig, och som primärnycklar med streck under sig.
Samma tillvägagångssätt användes sedan på ingår och finns.

Då filmex, hylla och uthyrning alla är svaga entiteter bearbetades dessa enligt följande. Alla entitetens attribut skrevs in och sedan skrevs även primärnycklarna från de entiteter som hjälpte till att identifiera entiteten in i parentesen. De attribut/nycklar som kom ifrån andra entiteter var ju foreign keys så dessa fick ett streck ovanför sig. Därefter fick entitetens primärnyckel och dess foreign keys även en understrykning för att markera att alla dessa krävdes för att identifiera entiteten.

Resultatet blev följande:

**Videobutik** (namn, adress, tel)

**Utgivare** (namn, tel)

**Kund** (p-nr, namn, tel, omdöme)

**Inkassofirma** (tel, namn, stad)

**Anställd** (a-nr, namn, lön, chef )

**Kampanj** (namn, vecko-nr, a-nr )

**Jobbar** ( butiksnamn , a-nr )

**Film** (titel, längd, kategori, utgivar-namn )

**Film-ex** (ex-nr, typ, pris, butiksnamn , titel )

**Ingår** ( kampanj-namn , filmtitel )

**Hylla** (hyll-nr, butiksnamn )

**Finns** ( hyll-nr, butiksnamn , ex-nr, ägarbutik, filmtitel )

**Uthyrning** (ut-datum, åter-datum, tillbaka, p-nr , ex-nr, ägarbutik, filmtitel , inkasso-tel )

# 3 Skapandet av databasen

När all modellering var avklarad var det så äntligen dags att formulera relationsdatamodellerna i språket SQL och skriva in dem i en databas. Detta kapitel går igenom processen steg för steg.

## 3.1 Grunden

Grunden var att först av allt skriva kommandot drop database a11emmjo; följt av create database a11emmjo; och use a11emmjo; Detta skapade själva grunden som krävdes för att kunna skapa databasen och sedan kunna fastställa databasens utseende och skriva in dess innehåll.

För att sedan skapa en tabell började man med att skriva create table följt av namnet som tabellen skulle heta. Därefter skrev man () ENGINE=INNODB; Innanför parentesen angav man sedan alla de attribut och eventuella foreign keys som man hade placerat innanför parentesen i relationsdatamodelleringen för entiteten. Varje attribut skrevs först ut med namn följt av vilken typ av data som attributet skulle vara. Om det exempelvis skulle vara en textsträng på 10 tecken skrev man CHAR(10). Attributen separerades sedan med ett komma. Efter attributen, fortfarande inom parentesen, skrev man primary key() och innanför den parentesen angav man vilka utav attributen som skulle vara entitetens nycklar, dvs. radens unika identifikation. Om entiteten även krävde foreign keys angav man även detta enlig följande: foreign key(A) references B(C). På A's plats skrev man in det attribut i den nuvarande tabellen som skulle syfta på och därmed inehålla en foreign key, på B's plats skrev man från vilken annan entitet/tabell som attributet skulle plockas in och därmed kopplas samman med. På C's plats skrev man till sist in vad attributet hade för namn i B's tabell.

## 3.2 Formulering av tabeller i databasen

När grunden var avklarad skrevs alla relationer, som tagits fram i relationsdatamodelleringen, in i databasen som tabeller enligt den syntax som beskrivs i 3.1. Se Bilaga B.

### 3.2.1 Videobutik

Videobutikens namn och adress fick bli en VARCHAR på 30 tecken. VARCHAR eftersom man omöjligt kunde fastställa antal tecken som strängen skulle bestå av, samtidigt som man ville spara på minnesutrymmet. 30 tecken valdes eftersom det kändes som en rimlig siffra för hur långt en adress alternativt ett namn kunde kräva. Även tel fick bli en VARCHAR och inte en INT trots att det handlar om siffror. Detta för att telefonnumret ju aldrig skulle hanteras som en sifferkombination och eventuellt utföras matematiska formler på, utan det skulle vara just en teckensträng. Dessutom var tanken att dela upp riktnumret och lokalnumret med ett bindestreck. Då telefonnummer kan variera i antal siffror och möjligheten att dessutom infoga ett bindestreck skulle finnas, så valdes 15 tecken.

### 3.2.2 Utgivare

Namn och tel behandlades här på samma sätt som tidigare.

### 3.2.3 Kund

Namn och tel behandlades på samma sätt som tidigare, samt att pNr sattes till att vara en CHAR på elva tecken. Detta eftersom ett korrekt personnummer alltid består av sex siffror följt av ett bindestreck och ytterligare fyra siffror. Totalt elva tecken. Dessutom sattes omdöme till VARCHAR på 150 tecken då tanken var att en kund skulle kunna ha ett detaljerat omdöme.

### 3.2.4 Inkassofirma

Tel och namn behandlades på samma sätt som tidigare. Dock lades uttrycket NOT NULL till på namn för att se till så att databasen krävde att ett namn skrevs in tillsammans med nyckeln tel för varje ny rad i denna tabell. Stad fick bli en VARCHAR på 20 tecken då det kändes rimligt.

### 3.2.5 Anställd

Namn behandlades som tidigare och sattes dessutom till NOT NULL. aNr blev däremot av typen CHAR då man visste från start att det alltid skulle innehålla ett anställningsnummer på exakt fem tecken. Även chef blev en CHAR på fem tecken eftersom den ju skulle referera till och därmed innehålla precis samma sak som aNr. Till sist blev lönen en INT på fem tecken. Detta eftersom man skulle kunna räkna med lönen och till exempel eventuellt ge en procentuell löneförhöjning. Fem tecken kändes som en rimlig siffra då ingen människa som jobbar på en videobutik rimligtvis bör kunna tjäna mer än 99 999:-

### 3.2.6 Kampanj
Namn behandlades som tidigare. aNr behandlades även detta som tidigare då det ju, precis som chef, skulle referera till den anställdes anställningsnummer och därmed innehålla samma sak. Veckonumret blev en INT på två tecken eftersom man eventuellt skulle vilja kunna räkna på hur många veckor en kampanj sträckte sig. Antalet två var för att vi har max 52 veckor på ett år.

### 3.2.7 Jobbar
Då butiksnamn hänvisar till namn i tabellen butik så fick den bli av samma typ. Samma sak gällde för aNr som hänvisar till aNr i tabellen anställd.

### 3.2.8 Film
Titel och kategori fick bägge bli VARCHAR med rimligt antal tecken för vardera. Eftersom utgivarNamn hänvisar till namn i tabellen utgivare fick den bli av samma typ och antal tecken. Längd fick tillsist bli en INT på tre tecken. Detta om man eventuellt skulle vilja jämföra hur många minuter längre en film är än en annan, eller liknande. Tre som antal tecken borde rimligtvis räcka då en film på 999 minuter skulle bli över 16 timmar lång, vilket inte känns särskilt rimligt.

### 3.2.9 FilmEx
Då butiksnamn och titel hänvisar till data i andra tabeller fick dessa bli av samma typ som det som det refererar till. Exemplarsnumret fick bli en CHAR på tre tecken eftersom det inte skulle gå att räkna med numret och det alltid skulle bestå av en sifferkombination på tre tecken. Eftersom typ syftar på om exemplaret är i formaten VHS, DVD eller Blu-Ray och namnet Blu-Ray är en kombination på sju tecken kändes det rimligt att sätta detta till VARCHAR(7). Pris sattes till sist till en INT på två tecken då man skulle kunna räkna ihop totalpriset på exempelvis ett antal filmer som ingick i en uthyrning. Det kändes inte heller rimligt att en film skulle kosta mer än 99:- att hyra.

### 3.2.10 Ingår
Eftersom kampanjnamn och filmtitel refererar till andra tabeller fick de båda samma typ som det de refererar till.

### 3.2.11 Hylla
Hyllnr fick bli en CHAR på tre tecken eftersom det alltid kommer att vara en sifferkombination på tre tecken som aldrig ska räknas med. Då butiksnamn hänvisar till butiken som hyllan står i och därmed tabellen videobutik så blir den av samma typ som det den refererar till, dvs. namn i tabellen videobutik.

### 3.2.12 Finns
Då alla attribut/kolumner i denna tabell hänvisar till attribut i andra tabeller så får de alla bli av samma typ som de attribut som de hänvisar till.

### 3.2.13 Uthyrning
Då utDatum, återDatum och tillbaka alla ska innehålla ett datum så fick de bli av typen datetime. Detta för att man sedan kan hantera datum och jämföra dessa med varandra om det alla är i detta format. Återdatumet fick dessutom bli NOT NULL eftersom en uthyrning aldrig skulle kunna genomföras utan att ett återdatum samtidigt skulle sättas. Övriga attribut hänvisade alla till attribut i andra tabeller och därför fick de alla samma typ som det attribut de hänvisade till.

# 4 Populering av databasen
När databasen väl var formulerad var det dags att fylla den med data. Varje punkt som listades i uppgiftsbeskrivningen (se Bilaga C) skrevs in i tur och ordning enligt följande syntax: insert into följt av namnet på den tabell som skulle fyllas. Därefter skrevs i en parentes alla tabellens attribut separerade med ett komma. Detta följdes av values() och i

denna andra parentes skrevs värdet för varje attribut för den unika raden in i samma ordning som attributen hade uppgivits i den första parentesen. Textsträngar omgavs av enkelfnuttar, medan int-värden och NULL skrevs in utan enkelfnuttar runt sig.
När alla punkter i Bilaga C var uppfyllda fylldes varje rad på med ytterligare data som fick diktas ihop allt eftersom för att göra varje rad komplett och därmed brukbar.
Till sist fylldes varje tabell på med ytterligare några helt påhittade rader efter egen förmåga för att underlätta den senare frågeoperationsprocessen. Detta för att kunna avgöra om frågeoperationerna gav rätt resultat.
Resultatet av det hela kan ses i Bilaga D.

# 5 Formulering av frågeoperationer

Det sista steget i uppgiften var att skriva specificerade frågeoperationer på den databas som just skapats. I detta kapitel tas varje fråga upp i tur och ordning tillsammans med en diskussion runt lösningen och den faktiska lösningen kan sedan ses i Bilaga E.

## 5.1 Hämta staden som inkassofirman med nummer 00326573892 kommer ifrån.

Detta plockas fram genom att välja att lista kolumnen stad i tabellen inkassofirma där kolumnen tel i samma rad har datan '00326-573892'. Observera att ett bindestreck har tillfogats telefonnumret då det är på detta sätt som datan har skrivits in vid populationen av databasen.

## 5.2 Hämta namn på videobutiken som den anställde med anställningsnummer 31366 arbetar på.

Med samma metod som i uppgift 5.1 listas nu kolumnen butiksnamn i tabellen jobbar där kolumnen aNr i samma rad har datan '31366'.

## 5.3 Hämta namn och telefonnummer på den utgivare som ger ut filmen Sigges Sommar.

Detta plockas fram genom att välja att lista kolumnerna namn och tel från utgivare i den kartesiska produkt/tabell som skapas när man joinar tabellerna utgivare och film. Villkoret för att en rad skall listas är att på en och samma rad skall kolumnen titel från film har datan 'Sigges Sommar' och sedan skall dessutom utgivarNamn från film ha identisk data med namn från utgivare.

## 5.4 Hämta namnet på videobutiken där den som ansvarar för kampanjen Vinterruset arbetar.

Detta plockas fram genom att precis som i uppgift 5.3 joina två tabeller, i detta fall kampanj och jobbar, i en kartesisk produkt. Sedan skrivs butiksnamn från jobbar ut om aNr för kampanj och jobbar är identiskt och namn från kampanj har datan 'Vinterruset' på samma rad.

## 5.5 Hämta namnet på kampanjen där filmen Hårda puckar ingår.

För att plocka fram detta listar man kolumnen kampanjnamn i tabellen ingar där kolumnen filmtitel på samma rad har datan 'Hårda puckar'. Observera att eftersom inte denna film ingår i någon kampanj i databasen kommer inget svar att ges. Frågan är dock kontrollerad med en annan film som villkor för att kontrollera dess riktighet.

## 5.6 Hämta filmexemplaren som kostar exakt lika mycket att hyra. (Tips: skapa instanser av samma tabell och jämför dess pris.)

För att plocka fram detta så måste man först skapa en dubblett utav tabellen filmEx. Detta åstadkoms genom att först ange namnet filmEx följt av ett mellanslag följt av det namn man vill att dubletten skall få, i detta fall temp. Därefter väljer man att plocka ut exNr och titel från filmEx där pris från filmEx är identiskt med pris från temp och dessutom inte exNr, butiksnamn och titel matchar och alltså är identiskt mellan filmEx och titel. Denna sista inte-sats krävs för att se till att inga identiska rader jämförs mot varandra då man faktiskt jämför

två identiska tabeller. Att just dessa tre inte skall vara identiska beror på att dessa i kombination är den unika nyckeln för varje rad i tabellen filmEx.

### 5.7 Hämta namn och personnummer för de kunder som inte lämnat åter sina filmer.

För att plocka fram detta måste man joina tabellerna uthyrning och kund och sedan kontrollera att tillbaka från uthyrning har datan NULL. Har den det så skrivs pNr och namn från kund på samma rad ut.

### 5.8 Hämta namn och personnummer för de kunder som inte lämnat tillbaka en film efter 10 dagar från uthyrningsdatum räknat.

Precis som i uppgift 5.7 måste man joina tabellerna uthyrning och kund. Därefter skrivs pNr och namn ut från kund om det på samma rad skiljer mer än tio dagar mellan tillbaka och utDatum från uthyrning. Detta kontrolleras med metoden DATEDIFF() som diskuteras på w3schools.com (2012d).

### 5.9 Visa alla kunder som inte hyrt någon film.

För att plocka fram detta måste man använda sig utav en NOT EXISTS() som innefattar ytterligare en fråga. Först tar man alltså fram en fråga som plockar fram alla rader (*) från uthyrning där pNr från uthyrning och kund är identiskt, dvs. man tar fram alla kunder som finns med i uthyrning och därmed har hyrt en film. För att sedan ta fram svaret på frågan listar man pNr och namn från kund där detta inte stämmer överens, dvs. alla de kunder som inte finns med i tabellen uthyrning. NOT EXISTS() diskuteras på TECH on the Net (2012a).

### 5.10 Lista de kunder som någon gång hyrt en film.

För att plocka fram alla de unika kunder som hyrt film, dvs. inte lista en kund flera gånger bara för att den hyrt film flera gånger, så används uttrycket DISTINCT. Med det sagt så listar man pNr och namn från kund efter att man joinat kund och uthyrning för att få fram svaret. Observera att då detta är en tolkningsfråga så har den tolkats som motsatsen till uppgift 5.9.

### 5.11 Lista de kunder som har hyrt exakt 2 filmer. (Tips: använd count.)

För att plocka fram detta måste man joina uthyrning och kund. Därefter grupperas pNr i uthyrning med hjälp av GROUP BY som diskuteras på w3schools.com (2012e). Därefter används COUNT() som diskuteras på w3schools (2012b) för att räkna antalet förekomster i en grupp. Till sist används HAVING som diskuteras på w3Schools.com (2012f) för att kontrollera om påståendet att en grupp av ett visst pNr i uthyrning har exakt två förekomster Om detta villkor stämmer så listas pNr och namn från kund.

### 5.12 Lista alla anställda och sortera namnet i omvänd ordning (Z först, A sist.)

Här listas alla med hjälp av * från anstalld och med hjälp av ORDER BY och DESC som diskuteras på tizag.com (2012b) så kan man sortera alla namn efter bokstavsordning från Z till A.

### 5.13 Hämta det genomsnittliga uthyrningspriset för samtliga uthyrningsexemplar.

Här listas de genomsnittliga priset från kolumnen pris i filmEx med hjälp av AVG() som diskuteras på w3schools.com (2012a).

### 5.14 Hämta medellönen för de anställda på respektive videobutik. (Tips: använd Group by.)

För att plocka fram detta måste man joina anställd och jobbar. Därefter använder man GROUP BY() på butiksnamnet från jobbar. Till sist listar man butiksnamn från jobbar och medellönen med hjälp av AVG() från anställd, på varje grupp.

### 5.15 Hämta medelkostnaden för varje kund som lämnat åter samtliga av sina filmer. (Tips: använd Not exists och is null.)

För att ta fram detta måste man först joina uthyrning och kund och sedan plocka fram alla rader från uthyrning där tillbaka har värdet NULL i en inre parentes. Detta placeras sedan efter en NOT EXISTS som resulterar i att enbart kunder som har lämnat tillbaka alla sina

filmer behandlas. Därefter joinas uthyrning med filmEx och man grupperar de kvarvarande raderna som uppfyller villkoren med GROUP BY(). Till sist plockar man ut pNr och namn från kund och medelpriset på pris från filmEx med hjälp av AVG().

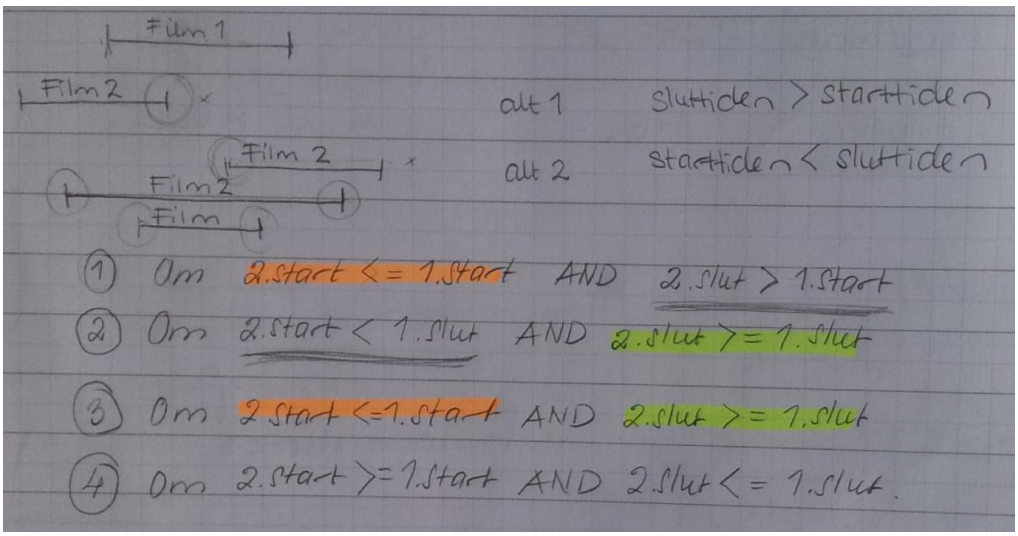
**5.16 Hämta telefonnumret för de inkassofirmor som krävt in filmer från lika många uthyrningar som inkassofirman med telefonnummer 00326573892. (Tips: använd select count(*) = select count(*)).**

För att plocka ut detta måste man skapa tre stycken frågeställningar. I den första inre använder man sig utav COUNT() för att plocka fram antal förekomster i uthyrning där inkassotel är '00326-573892'. Därefter skapar man den andra inre och räknar återigen antal förekomster med hjälp av COUNT() men nu joinar man inkassofirma och uthyrning och räknar antal rader som inte har det efterfrågade telefonnumret. Dessa jämförs sedan mot varandra och om de får samma resultat så är villkoret sant och i så fall så listas namn och tel från inkassofirma.

**5.17 Tanken är att ett filmexemplar inte kan bli uthyrd vid två tillfällen under samma tidsintervall. Undersök därför om det finns tidsintervall där ett filmexemplar har blivit uthyrt flera gånger under samma tidsintervall.**

Då tipset till denna uppgift var att använda sig utav en tempmapp så tittades det först tillbaka på uppgift 5.6 och konstaterades att man först och främst måste använda sig utav NOT() för att plocka bort jämförelsen mellan identiska rader när man använder sig utav en tempmapp. Därefter ritades Figur 1 upp enligt tips från kursföreläsare Christian Lennerholt för att underlätta tänkandet. Sedan skrevs de fyra olika alternativen, som skulle kunna resultera i att en uthyrning överlappar en annan, in i en parentes med OR emelllan för att tala om att ett utav alternativen skulle gälla. OR diskuteras på tizag.com (2012a) Till sist lades även en kontroll av att exNr och ägarbutik in. Detta för att se till att resultatet skulle peka på en och samma film tack vare kontrollen av dess nycklar. Om alla villkor skulle uppfyllas listades till sist all information på denna rad från uthyrning.

**Figur 1**




**5.18 Hämta all information om inkassofirmor som kommer från en stad som börjar på "E".**

För att plocka fram detta måste man använda sig utav funktionen LIKE och wildcardet % som lägger till noll till flera tecken efter E. LIKE-funktionen diskuteras på w3schools.com (2012h). Om stad från inkassofirma innehåller en textsträng som börjar med E och följs av noll till flera tecken så skrivs alla rader ut från inkassofirma där detta stämmer.

### 5.19 Hämta namn och adress för de kunder som angivit ett personnummer som inte är på formen XXXXXXXXXX där X är en siffra mellan 0 och 9. (Tips: använd rlike eller regxp.)

För att plocka fram detta måste man använda sig utav funktionen REGEXP() som diskuteras på GO4EXPERT (2012). Man börjar alltså med att kontrollera om strängen pNr från kund enbart innehåller siffror med hjälp av REGEXP() och tecknen ^(som betyder startar med), +(som betyder innehåller), $(som betyder slutar med) och [ ] (som omger det man frågar efter). Därefter behöver man använda LENGTH() som diskuteras på 1KD (2012) för att kontrollera att längden på textsträngen är exakt tio tecken lång. Till sist placeras detta i en NOT() och om det finns några rader i kund där dessa villkor stämmer överens så listas namn och tel från kund. Observera att frågan efterfrågar adress men då databasen inte innehåller detta har friheten tagits att byta ut detta mot tel i stället.

### 5.20 Hämta namn och telefonnummer för den kund som har hyrt den dyraste filmen. (Tips: använd max().)

Här använder man sig utav MAX() som diskuteras på TECH on the Net (2012b) för att plocka fram den dyraste filmen i filmEx. Eftersom den dessutom måste ha blivit uthyrd joinar man filmEx med uthyrning. Detta placeras sedan i en inre parentes som sedan jämförs mot det pris i filmEx som listas om man joinar uthyrning, filmEx och kund. Därefter listas den kund som har hyrt den dyraste filmen med namn och tel om det finns någon.

### 5.21 Hämta namnet på kunden som senast hyrde filmen Sataythai på Vipp-video.

För att plocka fram detta måste man först plocka fram det senaste utDatumet från uthyrning i en inre parentes. Därefter joinar man kund och uthyrning, kontrollerar att filmens titel är den efterfrågade och att utDatumet är samma som det senaste utDatumet. Därefter listas den berörda kundens namn. Observera då det inte finns en film inlagd i databasen som heter Sataythai kontrollerades frågans riktighet istället mot andra titlar.

### 5.22 Lista de uthyrningar som skett den senaste veckan. (Tips: använd curdate() eller liknande.)

Här använder man sig av utav DATE_SUB() som diskuteras på w3schools.com (2012c) och drar av ett intervall av sju dagar från CURDATE(), dvs. dagens datum och sedan jämför man detta mot utDatum från uthyrning och om utDatumet är större innebär det att en uthyrning har skett den senaste veckan. Om så är fallet listas alla dessa rader.

### 5.23 Höj lönen för alla anställda med en månadslön mellan 10000 och 12000 kr med 22%.

För att åstadkomma detta använder man sig utav UPDATE för att tala om i vilken tabell det ska ändras och SET för vad som ska hända. I detta fall skall lön från anställd bli lön*1,22, dvs. 22% högre. Men det skall bara ändras på rader i anställd där lön är antingen är högre eller lika med 10 000:- och mindre än eller lika med 12 000:- Om SET och UPDATE kan man läsa på w3schools.com (2012g).

### 5.24 Ta bort exemplaret med nummer 442.

För att lösa detta skulle man kunna använda sig utav DELETE FROM på tabellen filmEx och specificera att det ska vara exemplaret med nr '442'. Detta kommer dock inte att fungera eftersom att exNr som i detta fall är en primärnyckel, även är en foreignkey i bl.a. tabellen finns. För att det skall fungera måste alla hänvisningar till det som skall raderas, plockas bort. Detta innebär att denna uppgift inte egentligen är löst. DELETE FROM diskuteras på tutorialpoint (2012).

### 5.25 Ta bort den anställde med anställningsnummer 31366.

Precis som i uppgift 5.24 kommer detta inte vara möjligt då aNr som i detta fall är en primärnyckel, även är en foreignkey i samma tabell då den hänvisar till den chef som den anställde har. Därav är denna uppgift egentligen inte heller löst.

## Referenser


1KD (2012) SQL Length Function. Tillgänglig på Internet: http://www.1keydata.com/sql/sql-length.html [Hämtad 12.05.15].

GO4EXPERT (2012) Regular Expressions in MySQL. Tillgänglig på Internet: http://www.go4expert.com/forums/showthread.php?t=2337 [Hämtad 12.05.15].

TECH on the Net (2012a) SQL: EXISTS Condition. Tillgänglig på Internet: http://www.techonthenet.com/sql/exists.php [Hämtad 12.05.01].

TECH on the Net (2012b) SQL: MAX Function. Tillgänglig på Internet: http://www.techonthenet.com/sql/max.php [Hämtad 12.05.15].

tizag.com (2012a) sql - or. Tillgänglig på Internet: http://www.tizag.com/sqlTutorial/sqlandor.php [Hämtad 12.05.17].

tizag.com (2012b) sql - order by. Tillgänglig på Internet: http://www.tizag.com/sqlTutorial/sqlorderby.php [Hämtad 12.05.08].

tutorialpoint (2012) MySQL DELETE Query. Tillgänglig på Internet: http://www.tutorialspoint.com/mysql/mysql-delete-query.htm [Hämtad 12.05.17].

w3schools (2012a) SQL AVG() Function. Tillgänglig på Internet: http://www.w3schools.com/sql/sql_func_avg.asp [Hämtad 12.05.08].

w3schools (2012b) SQL COUNT() Function. Tillgänglig på Internet: http://www.w3schools.com/sql/sql_func_count.asp [Hämtad 12.05.01].

w3schools (2012c) MySQL DATE_SUB() Function. Tillgänglig på Internet: http://www.w3schools.com/sql/func_date_sub.asp [Hämtad 12.05.15].

w3schools (2012d) MySQL DATEDIFF() Function. Tillgänglig på Internet: http://www.w3schools.com/sql/func_datediff_mysql.asp [Hämtad 12.05.01].

w3schools (2012e) SQL GROUP BY Statement. Tillgänglig på Internet: http://www.w3schools.com/sql/sql_groupby.asp [Hämtad 12.05.01].

w3schools (2012f) SQL HAVING Clause. Tillgänglig på Internet: http://www.w3schools.com/sql/sql_having.asp [Hämtad 12.05.01].

w3schools (2012g) SQL UPDATE Statement. Tillgänglig på Internet: http://www.w3schools.com/sql/sql_update.asp [Hämtad 12.05.15].

w3schools (2012h) SQL Wildcards. Tillgänglig på Internet: http://www.w3schools.com/sql/sql_wildcards.asp [Hämtad 12.05.15].

# Bilaga A. Objekt och villkor som skulle överföras till ER-modellen

**Videobutik:** En videobutik identifieras av sitt namn och har även en adress och ett telefonnummer. En butik måsta ha minst en anställd. En butik kan ha ett flertal hyllor och inneha ett flertal filmexemplar.

**Anställd:** En anställd identifieras av sitt anställningsnummer och har även ett namn och en lön. En anställd kan ha en chef. En chef kan vara chef för flera anställda. En anställd kan arbeta i flera butiker och kan ansvara för flera kampanjer.

**Film:** En film identifieras av sin titel och har även en längd samt tillhör en viss kategori. En film kan ingå i flera kampanjer men ges alltid ut av en enda utgivare. Varje film kan finnas i flera filmexempelar.

**Utgivare:** En utgivare kan ge ut flera filmer. En utgivare identifieras av sitt namn och har även ett telefonnummer.

**Kampanj:** En kampanj identifieras av sitt namn och har även ett veckonummer. En kampanj kan ha en anställd som ansvarar för kampanjen och varje kampanj kan omfatta ett flertal olika filmer.

**Hylla:** En hylla är unik för den butik där hyllan finns placerad och identifieras därmed av ett hyllnummer samt den butik där hyllan står. I en hylla kan det såklart finnas flera filmexemplar av flera filmer.

**Filmexemplar:** Ett filmexemplar är unikt för en butik samt för en film. Själva filmexemplaret identifieras av ett nummer men i en och samma butik kan det finnas flera lika nummer, men inte för samma film. Varje filmexemplar har ett pris och är av en viss typ, exempelvis DVD, VHS eller Bluray. Ett filmexemplar kan hyras ut flera gånger. Filmexemplar kan finnas i olika hyllor, alltså kan filmexemplar 1 finnas i en hylla och filmexemplar 2 i en annan hylla.

**Uthyrning:** En uthyrning är unik för varje filmexemplar och för varje kund. Själva uthyrningen identifieras partiellt av uthyrningstid. En uthyrning har ett senast återlämningsdatum och ett verkligt återlämningsdatum som visar när filmen faktiskt återlämnats. Datum är i detta sammanhang synonymt med en unik tidpunkt, det vill säga datum tillsammans med klockslag. Varje uthyrning måste även ha en inkassofirma knuten till sig. Inkassofirman driver in filmer som inte återlämnats i tid.

**Kund:** En kund identifieras unikt av personnummer och har även ett namn, ett telefonnummer samt ett omdöme. En kund kan vara inblandad i flera uthyrningar.

**Inkassofirma:** En inkassofirma identifieras av sitt telefonnummer och kan vara inblandad i flera uthyrningar. En inkassofirma har dessutom en stadstillhörighet samt ett namn.

# Bilaga B. **Skapandet av databasen**

```
1  drop database a11emmjo;
2  create database a11emmjo;
3  use a11emmjo;
4
5  CREATE TABLE videobutik(
6    namn VARCHAR(30),
7    adress VARCHAR(30),
8    tel VARCHAR(15),
9    PRIMARY KEY(namn)
10 ) ENGINE=INNODB;
11
12 CREATE TABLE utgivare(
13   namn VARCHAR(30),
14   tel VARCHAR(15),
15   PRIMARY KEY(namn)
16 ) ENGINE=INNODB;
17
18 CREATE TABLE kund(
19   pNr CHAR(11),
20   namn VARCHAR(30),
21   tel VARCHAR(15),
22   omdöme VARCHAR(150),
23   PRIMARY KEY(pNr)
24 ) ENGINE=INNODB;
25
26 CREATE TABLE inkassofirma(
27   tel VARCHAR(15),
28   namn VARCHAR(30) NOT NULL,
29   stad VARCHAR(20),
30   PRIMARY KEY(tel)
31 ) ENGINE=INNODB;
32
33 CREATE TABLE anstalld(
34   aNr CHAR(5),
35   namn VARCHAR(30) NOT NULL,
36   lon INT(5),
37   chef CHAR(5), --Denna måste ha samma typ som aNr då de ska sammankopplas.
38   PRIMARY KEY(aNr),
39   FOREIGN KEY(chef) REFERENCES anstalld(aNr)
40 ) ENGINE=INNODB;
41
42 CREATE TABLE kampanj(
43   namn VARCHAR(30),
44   veckoNr INT(2),
45   aNr CHAR(5),
46   PRIMARY KEY(namn),
47   FOREIGN KEY(aNr) REFERENCES anstalld(aNr)
48 ) ENGINE=INNODB;
49
50 CREATE TABLE jobbar(
51   butiksnamn VARCHAR(30),
52   aNr CHAR(5),
53   PRIMARY KEY(butiksnamn,aNr),
54   FOREIGN KEY(butiksnamn) REFERENCES videobutik(namn),
55   FOREIGN KEY(aNr) REFERENCES anstalld(aNr)
56 ) ENGINE=INNODB;
57
58 CREATE TABLE film(
59   titel VARCHAR(50),
```

```
60      langd INT(3),
61      kategori VARCHAR(20),
62      utgivarNamn VARCHAR(30),
63      PRIMARY KEY(titel),
64      FOREIGN KEY(utgivarNamn) REFERENCES utgivare(namn)
65    ) ENGINE=INNODB;
66
67    CREATE TABLE filmEx(
68      exNr CHAR(3),
69      typ VARCHAR(7),
70      pris INT(2),
71      butiksnamn VARCHAR(30),
72      titel VARCHAR(50),
73      PRIMARY KEY(exNr,butiksnamn,titel),
74      FOREIGN KEY(butiksnamn) REFERENCES videobutik(namn),
75      FOREIGN KEY(titel) REFERENCES film(titel)
76    ) ENGINE=INNODB;
77
78    CREATE TABLE ingar(
79      kampanjnamn VARCHAR(30),
80      filmtitel VARCHAR(50),
81      PRIMARY KEY(kampanjnamn,filmtitel),
82      FOREIGN KEY(kampanjnamn) REFERENCES kampanj(namn),
83      FOREIGN KEY(filmtitel) REFERENCES film(titel)
84    ) ENGINE=INNODB;
85
86    CREATE TABLE hylla(
87      hyllNr CHAR(3),
88      butiksnamn VARCHAR(30),
89      PRIMARY KEY(hyllNr,butiksnamn),
90      FOREIGN KEY(butiksnamn) REFERENCES videobutik(namn)
91    ) ENGINE=INNODB;
92
93    CREATE TABLE finns(
94      hyllNr CHAR(3),
95      butiksnamn VARCHAR(30),
96      exNr CHAR(3),
97      agarbutik VARCHAR(30),
98      filmtitel VARCHAR(50),
99      PRIMARY KEY(hyllNr,butiksnamn,exNr,agarbutik,filmtitel),
100     FOREIGN KEY(hyllNr,butiksnamn) REFERENCES hylla(hyllNr,butiksnamn),
101     FOREIGN KEY(exNr,agarbutik,filmtitel) REFERENCES
102   filmEx(exNr,butiksnamn,titel)
103   ) ENGINE=INNODB;
104
105   CREATE TABLE uthyrning(
106     utDatum datetime,
107     aterDatum datetime NOT NULL,
108     tillbaka datetime,
109     pNr VARCHAR(11),
110     exNr CHAR(3),
111     agarbutik VARCHAR(30),
112     filmtitel VARCHAR(50),
113     inkassoTel VARCHAR(15),
114     PRIMARY KEY(utDatum,pNr,exNr,agarbutik,filmtitel,inkassoTel),
115     FOREIGN KEY(pNr) REFERENCES kund(pNr),
116     FOREIGN KEY(exNr,agarbutik,filmtitel) REFERENCES
117   filmEx(exNr,butiksnamn,titel),
118     FOREIGN KEY(inkassoTel) REFERENCES inkassofirma(tel)
119   ) ENGINE=INNODB;
```

## Bilaga C. **Punkter som krävdes vid populationen av databasen**

- Videobutikerna med namnet Vipp-video och Axvallsgrillen.
- Chefen Bosse Lingqvist med anställningsnummer 31366 och som arbetar på Vippvideo och har 36000kr i månadslön.
- Den anställde Biff Lindström som arbetar på Vipp-video och har Bosse som chef och tjänar 8420kr i månaden. Biffs anställningsnummer är 87231.
- Angela Bly som är chef på Axvallsgrillen och som tjänar 23000kr i månaden. Anställningsnumret är 38065.
- Ulf Bertilsson med anställningsnummer 66633 arbetar på Axvallsgrillen och har Angela som chef.
- Kampanjen Vinterruset som ansvaras av Ulf och startar vecka 48.
- Biff ansvarar för kampanjen Sommarrullen som startar vecka 26.
- Utgivarna Östproduktion och Vaniljrelease med telefonnummer 097854321 och 068723472.
- Filmen Sigges Sommar är 97min lång och ges ut av Östproduktion och tillhör kategorin romantik. Den finns i 5st exemplar där exemplarnumren 111, 122 och 133 tillhör typen DVD och kostar 48kr att hyra. Exemplarnumren 442 och 443 tillhör typen Bluray och kostar 78kr att hyra.
- Filmen Hårda puckar och är 62min lång och ges ut av Östproduktion och tillhör kategorin sport. Filmen finns bara i ett exemplar i typen VHS. Exemplarnumret är 671 och hyrpriset är 29kr.
- Hylla 114 som finns i Axvallsgrillen och lagrar DVD-filmerna Sigges Sommar.
- Hylla 115 lagrar alla Bluray-filmerna av Sigges Sommar.
- I butiken Vipp-video står hyllan med nummer 114 och som lagrar 3st exemplar av DVD-actionfilmen Mopedjakten.
- Kunden Josse Björk med personnummer 920128-5884 och har omdömet pålitlig.
- Björn Bark med telefonnumret 0788883212 och personnummer 670723-5835 har omdömet stamkund.
- Inkassosfirman Inkasso AB från Stöpen har telefonnumret 053334577.
- Inkassofirman Grå&Trist HB från Varola har telefonnumret 00326573892.
- Uthyrningen den 12/1-2012 kl1801 inkluderade filmexemplaret 442 och hyrdes av Josse Björk. Filmen återlämnades den 13/1-2012 kl1759. Inkassofirman Grå&Trist HB övervakade uthyrningen.
- Uthyrningen den 16/2-2012 kl1234 inkluderade filmexemplaret 671 och hyrdes av Björn Bark. Filmen skulle vara återlämnad den 20/2 kl1800. Inkassofirman Inkasso AB driver in filmen.
- Uthyrningen den 12/1-2012 kl1700 inkluderade filmexemplaret 442 och hyrdes av Björn Bark. Filmen återlämnades den 13/1-2012 kl1759. Inkassofirman Grå&Trist HB övervakade uthyrningen. (Avser en felaktighet då samma exemplar har hyrts ut till två olika kunder).

# Bilaga D. **Populering av databasen**

```
insert into videobutik(namn,adress,tel) values('Vipp-video','Skaragatan 5,
Lidköping','0510-12345');
insert into videobutik(namn,adress,tel) values('Axvallsgrillen','Granstigen
7, Axvall','0511-14745');
insert into videobutik(namn,adress,tel) values('Åkes Video','Marumsgatan
23, Skara','0511-27345');
insert into videobutik(namn,adress,tel) values('All-Video','Källegatan 2,
Skara','0511-54115');
insert into videobutik(namn,adress,tel) values('Hemmavideo','Storgatan 33,
Skövde','0500-50045');

insert into anstalld(aNr,namn,lon,chef) values('31366','Bosse
Lindqvist','36000',NULL); -- Chef på Vipp-video.
insert into anstalld(aNr,namn,lon,chef) values('87231','Biff
Lindström','8420','31366');
insert into anstalld(aNr,namn,lon,chef) values('38065','Angela
Bly','23000',NULL); -- Chef på Axvallsgrillen.
insert into anstalld(aNr,namn,lon,chef) values('66633','Ulf
Bertilsson','13465','38065');
insert into anstalld(aNr,namn,lon,chef) values('57440','Annika
Bengtsson','34200',NULL); -- Chef på Hemmavideo
insert into anstalld(aNr,namn,lon,chef) values('35440','Robert
Persson','28100',NULL); -- Chef på All-Video
insert into anstalld(aNr,namn,lon,chef) values('21157','Åke
Edvinsson','21700',NULL); -- Chef på Åkes Video
insert into anstalld(aNr,namn,lon,chef) values('21158','Test
Testsson','11500',38065);

insert into jobbar(butiksnamn,aNr) values('Vipp-video','31366'); -- Bosse
Lindqvist
insert into jobbar(butiksnamn,aNr) values('Vipp-video','87231'); -- Biff
Lindström
insert into jobbar(butiksnamn,aNr) values('Axvallsgrillen','38065'); --
Angela Bly
insert into jobbar(butiksnamn,aNr) values('Axvallsgrillen','66633'); -- Ulf
Bertilsson
insert into jobbar(butiksnamn,aNr) values('Hemmavideo','57440'); -- Annika
Bengtsson
insert into jobbar(butiksnamn,aNr) values('All-Video','35440');  -- Robert
Persson
insert into jobbar(butiksnamn,aNr) values('Åkes Video','21157'); -- Åke
Edvinsson

insert into kampanj(namn,veckoNr,aNr) values('Vinterruset','48','66633');
insert into kampanj(namn,veckoNr,aNr) values('Sommarrullen','26','87231');
insert into kampanj(namn,veckoNr,aNr) values('Halloween','44','21157');
insert into kampanj(namn,veckoNr,aNr) values('Påskägg','14','21157');
insert into kampanj(namn,veckoNr,aNr) values('Sweethearts','7','38065');

insert into utgivare(namn,tel) values('Östproduktion','0978-54321');
insert into utgivare(namn,tel) values('Vaniljrelease','0687-23472');
insert into utgivare(namn,tel) values('Film i Väst','0520-357400');
insert into utgivare(namn,tel) values('Svensk Film','08-353535');
insert into utgivare(namn,tel) values('Warner Bros','08-220000');

insert into film(titel,langd,kategori,utgivarNamn) values('Sigges
Sommar','97','romantik','Östproduktion');
```

```
insert into film(titel,langd,kategori,utgivarNamn) values('Hårda
puckar','62','sport','Östproduktion');
insert into film(titel,langd,kategori,utgivarNamn)
values('Mopedjakten','84','action','Vaniljrelease');
insert into film(titel,langd,kategori,utgivarNamn)
values('Festival','73','komedi','Film i Väst');
insert into film(titel,langd,kategori,utgivarNamn) values('Omen
III','95','skräck','Warner Bros');
insert into film(titel,langd,kategori,utgivarNamn) values('November
rain','81','drama','Warner Bros');
insert into film(titel,langd,kategori,utgivarNamn) values('Pippi Långstrump
på de sju haven','53','tecknat','Svensk Film');

insert into filmEx(exNr,typ,pris,butiksnamn,titel)
values('111','DVD','48','Axvallsgrillen','Sigges Sommar');
insert into filmEx(exNr,typ,pris,butiksnamn,titel)
values('122','DVD','48','Axvallsgrillen','Sigges Sommar');
insert into filmEx(exNr,typ,pris,butiksnamn,titel)
values('133','DVD','48','Axvallsgrillen','Sigges Sommar');
insert into filmEx(exNr,typ,pris,butiksnamn,titel)
values('442','Bluray','78','Axvallsgrillen','Sigges Sommar');
insert into filmEx(exNr,typ,pris,butiksnamn,titel)
values('443','Bluray','78','Axvallsgrillen','Sigges Sommar');
insert into filmEx(exNr,typ,pris,butiksnamn,titel)
values('671','VHS','29','Vipp-video','Hårda puckar');
insert into filmEx(exNr,typ,pris,butiksnamn,titel)
values('900','DVD','35','Vipp-video','Mopedjakten');
insert into filmEx(exNr,typ,pris,butiksnamn,titel)
values('901','DVD','35','Vipp-video','Mopedjakten');
insert into filmEx(exNr,typ,pris,butiksnamn,titel)
values('902','DVD','35','Vipp-video','Mopedjakten');

insert into hylla(hyllNr,butiksnamn) values('114','Axvallsgrillen');
insert into hylla(hyllNr,butiksnamn) values('115','Axvallsgrillen');
insert into hylla(hyllNr,butiksnamn) values('114','Vipp-video');

insert into finns(hyllNr,butiksnamn,exNr,agarbutik,filmtitel)
values('114','Axvallsgrillen','111','Axvallsgrillen','Sigges Sommar');
insert into finns(hyllNr,butiksnamn,exNr,agarbutik,filmtitel)
values('114','Axvallsgrillen','122','Axvallsgrillen','Sigges Sommar');
insert into finns(hyllNr,butiksnamn,exNr,agarbutik,filmtitel)
values('114','Axvallsgrillen','133','Axvallsgrillen','Sigges Sommar');
insert into finns(hyllNr,butiksnamn,exNr,agarbutik,filmtitel)
values('115','Axvallsgrillen','442','Axvallsgrillen','Sigges Sommar');
insert into finns(hyllNr,butiksnamn,exNr,agarbutik,filmtitel)
values('115','Axvallsgrillen','443','Axvallsgrillen','Sigges Sommar');
insert into finns(hyllNr,butiksnamn,exNr,agarbutik,filmtitel)
values('114','Vipp-video','900','Vipp-video','Mopedjakten');
insert into finns(hyllNr,butiksnamn,exNr,agarbutik,filmtitel)
values('114','Vipp-video','901','Vipp-video','Mopedjakten');
insert into finns(hyllNr,butiksnamn,exNr,agarbutik,filmtitel)
values('114','Vipp-video','902','Vipp-video','Mopedjakten');

insert into kund(pNr,namn,tel,omdöme) values('920128-5884','Josse
Björk',NULL,'Pålitlig');
insert into kund(pNr,namn,tel,omdöme) values('670723-5835','Björn
Bark','0788-883212','Stamkund');
insert into kund(pNr,namn,tel,omdöme) values('780419-5945','Karin
Andersson','0511-252142','Ofta sen');
insert into kund(pNr,namn,tel,omdöme) values('911025-6004','Matilda
Rosenqvist','0707-352565','Pålitlig');
```

```
insert into kund(pNr,namn,tel,omdöme) values('860619-3576','Nicklas
Bengtsson','0707-113145','Slarver');
insert into kund(pNr,namn,tel,omdöme) values('8112241234','Rudolf
Andersson','0707-571919','Ny');
insert into kund(pNr,namn,tel,omdöme) values('87112112345','Oscar
Bengtsson','0707-447192','Ny');

insert into inkassofirma(tel,namn,stad) values('0533-34577','Inkasso
AB','Stöpen');
insert into inkassofirma(tel,namn,stad) values('00326-573892','Grå &
Trist','Varola');
insert into inkassofirma(tel,namn,stad) values('0511-
77400','Torpeden','Skara');
insert into inkassofirma(tel,namn,stad) values('011-
220000','Help','Emmaboda');

insert into
uthyrning(utDatum,aterDatum,tillbaka,pNr,exNr,agarbutik,filmtitel,inkassoTe
l) values('2012-01-12 18:01:00','2012-01-13 18:00:00','2012-01-13
17:59:00','920128-5884','442','Axvallsgrillen','Sigges Sommar','00326-
573892');
insert into
uthyrning(utDatum,aterDatum,tillbaka,pNr,exNr,agarbutik,filmtitel,inkassoTe
l) values('2012-02-16 12:34:00','2012-02-20 18:00:00',NULL,'670723-
5835','671','Vipp-video','Hårda puckar','0533-34577');
insert into
uthyrning(utDatum,aterDatum,tillbaka,pNr,exNr,agarbutik,filmtitel,inkassoTe
l) values('2012-01-12 17:00:00','2012-01-16 18:00:00','2012-01-13
17:59:00','670723-5835','442','Axvallsgrillen','Sigges Sommar','00326-
573892'); -- Detta är inte en möjlig rad då filmen hyrs ut samtidigt som
ovan, utan ett medvetet fel!
insert into
uthyrning(utDatum,aterDatum,tillbaka,pNr,exNr,agarbutik,filmtitel,inkassoTe
l) values('2012-04-19 15:01:23','2012-04-24 18:00:00','2012-04-30
16:23:15','780419-5945','900','Vipp-video','Mopedjakten','0533-34577');
insert into
uthyrning(utDatum,aterDatum,tillbaka,pNr,exNr,agarbutik,filmtitel,inkassoTe
l) values('2012-05-08 19:57:12','2012-05-12 18:00:00','2012-05-10
12:13:27','860619-3576','902','Vipp-video','Mopedjakten','0511-77400');

insert into ingar(kampanjnamn,filmtitel) values('Sweethearts','Sigges
Sommar');
insert into ingar(kampanjnamn,filmtitel) values('Halloween','Omen III');
insert into ingar(kampanjnamn,filmtitel) values('Sommarrullen','Festival');
insert into ingar(kampanjnamn,filmtitel) values('Påskägg','Pippi Långstrump
på de sju haven');
insert into ingar(kampanjnamn,filmtitel)
values('Vinterruset','Mopedjakten');
```

# Bilaga E. **Frågeoperationerna**

```
1  -- 1. Hämta staden som inkassofirman med nummer 00326573892 kommer ifrån.
2
3  -- Detta tar jag reda på genom att kolla i tabellen som innehåller all info
4     om inkassofirmor och sedan kontrollerar värdet för attributet stad på
5     samma rad som värdet på attributet tel är 00326573892.
6
7  -- OBS! Då jag använt bindestreck i telnr i tabellen lägger jag till det i
8     frågan för att få ut mitt svar.
9
10    SELECT stad
11    FROM inkassofirma
12    WHERE tel='00326-573892';
13
14 -- 2. Hämta namn på videobutiken som den anställde med anställningsnummer
15 31366 arbetar på.
16
17 --  Detta tar jag reda på genom att kolla i tabellen jobbar och sedan
18 kontrollerar jag värdet för attributet butiksnamn på raden där attributet
19 aNr är 31366.
20
21    SELECT butiksnamn
22    FROM jobbar
23    WHERE aNr='31366';
24
25 -- 3. Hämta namn och telefonnummer på den utgivare som ger ut filmen Sigges
26 Sommar.
27
28 -- Detta tar jag reda på genom att JOINA tabellerna utgivare och film.
29
30    SELECT utgivare.namn, utgivare.tel
31    FROM utgivare, film
32    WHERE film.titel='Sigges Sommar' AND film.utgivarNamn=utgivare.namn;
33
34 -- 4. Hämta namnet på videobutiken där den som ansvarar för kampanjen
35 Vinterruset arbetar.
36
37 -- Först plockar jag fram raden i tabellen kampanj där namn är Vinterruset
38 och då får jag fram anställningsnr på den som är ansvarig. Sedan JOINAR jag
39 tabellerna kampanj och jobbar och får då fram raden som säger att samma
40 anställningsnr som tidigare jobbar på...
41
42    SELECT jobbar.butiksnamn
43    FROM kampanj,jobbar
44    WHERE kampanj.namn='Vinterruset' AND kampanj.aNr=jobbar.aNr;
45
46 -- 5. Hämta namnet på kampanjen där filmen Hårda puckar ingår.
47
48 -- För att få fram detta frågar jag tabellen ingar nedanstående fråga, men
49 eftersom Hårda puckar inte ingår i en kampanj kommer inget svar att ges.
50
51 -- OBS! Jag har dock testat med en annan film så att frågan fungerar.
52
53    SELECT kampanjnamn
54    FROM ingar
55    WHERE filmtitel='Hårda puckar';
56
57 -- 6. Hämta filmexemplaren som kostar exakt lika mycket att hyra. (Tips:
58 skapa instanser av samma tabell och jämför dess pris.)
```

```
59
60  -- För att få fram detta måste jag skapa en ny temptabell av filmEx och
61  sedan JOINA den med filmEx-tabellen. Därefter måste jag plocka fram alla
62  rader som har samma värde på attributet pris. Tillsist måste jag plocka
63  bort alla rader som har exakt samma nycklar. Eftersom nycklarna är unika
64  för sin rad tas alla dubletter mellan tabellerna bort.
65
66    SELECT filmEx.exNr,filmEx.titel
67    FROM filmEx,  filmEx temp
68    WHERE filmEx.pris=temp.pris AND NOT(filmEx.exNr=temp.exNr AND
69  filmEx.butiksnamn=temp.butiksnamn AND filmEx.titel=temp.titel);
70
71  -- 7. Hämta namn och personnummer för de kunder som inte lämnat åter sina
72  filmer.
73
74  -- Jag JOINAR tabellerna uthyrning och kund och de rader som innehåller ett
75  NULL-värde på attributet tillbaka skrivs sedan ut.
76
77    SELECT kund.pNr,kund.namn
78    FROM uthyrning,kund
79    WHERE uthyrning.tillbaka IS NULL AND uthyrning.pNr=kund.pNr;
80
81  -- 8. Hämta namn och personnummer för de kunder som inte lämnat tillbaka en
82  film efter 10 dagar från uthyrningsdatum räknat.
83
84  -- Precis som ovan JOINAR jag uthyrning och kund och tar sedan fram de
85  rader där det skiljer mer än 10 dgr mellan tillbakadatumet och
86  uthyrningsdatumet.
87
88    SELECT kund.pNr,kund.namn
89    FROM uthyrning,kund
90    WHERE DATEDIFF(uthyrning.tillbaka,uthyrning.utDatum)>10 AND
91  uthyrning.pNr=kund.pNr;
92
93  -- 9. Visa alla kunder som inte hyrt någon film.
94
95  -- Jag JOINAR tabellerna kund och uthyrning och plockar sedan ut alla rader
96  där pNr i uthyrning och kund inte matchar.
97
98    SELECT kund.pNr,kund.namn
99    FROM kund
100   WHERE NOT EXISTS(SELECT * FROM uthyrning WHERE uthyrning.pNr=kund.pNr);
101
102 -- 10. Lista de kunder som någon gång hyrt en film.
103
104 -- Jag JOINAR tabellerna kund och uthyrning och plockar sedan ut alla rader
105 där pNr i uthyrning och kund matchar. Eftersom Björn Bark kommer att listas
106 2 ggr iom att han har hyrt film 2 ggr så lägger jag till DISTINCT för att
107 ta bort dublettrader.
108
109 -- OBS! Då detta är en tolkningsfråga, tolkar jag den som motsatsen till
110 frågan ovan.
111
112   SELECT DISTINCT kund.pNr,kund.namn
113   FROM kund,uthyrning
114   WHERE uthyrning.pNr=kund.pNr;
115
116 -- 11. Lista de kunder som har hyrt exakt 2 filmer. (Tips: använd count.)
117
118 -- Först JOINAR jag tabellerna uthyrning och kund precis som ovan och sedan
119 grupperar jag varje rad med samma pNr i en grupp. Därefter kontrollerar jag
```

```
om gruppen innehåller 2 rader. Gör den det skrivs pNr och namn ut på den
person som ingår i gruppen.

   SELECT kund.pNr,kund.namn
   FROM uthyrning,kund
   WHERE kund.pNr=uthyrning.pNr
   GROUP BY uthyrning.pNr
   HAVING (COUNT(uthyrning.pNr)=2);

-- 12. Lista alla anställda och sortera namnet i omvänd ordning (Z först, A
sist.)

-- Jag skriver ut alla rader i tabellen anstalld (eftersom WHERE inte finns
med, sorteras inget bort) och listar sedan raderna sorterade i fallande
ordning (DESC istället ASC som är stigande) utifrån värdet i attributet
namn.

   SELECT *
   FROM anstalld
   ORDER BY namn DESC;

-- 13. Hämta det genomsnittliga uthyrningspriset för samtliga
uthyrningsexemplar.

-- Jag kör AVG() på pris som är det jag vill beräkna medelvärdet på i
tabellen filmEx.

   SELECT AVG(pris)
   FROM filmEx;

-- 14. Hämta meddellönen för de anställda på respektive videobutik. (Tips:
använd Group by.)

-- Jag JOINAR anstalld och jobbar och plockar ut de rader där anställd pch
jobbar har samma pNr. Jag grupperar sedan efter butiksnamnet och skriver ut
varje butik och dess anställdas medellön.

   SELECT jobbar.butiksnamn,AVG(anstalld.lon)
   FROM anstalld,jobbar
   WHERE anstalld.aNr=jobbar.aNr
   GROUP BY jobbar.butiksnamn;

-- 15. Hämta medelkostnaden för varje kund som lämnat åter samtliga av sina
filmer. (Tips: använd Not exists och is null.)

-- Först tar jag fram alla kunder som hyrt en film. Sedan alla som inte har
några NULL på återlämningstid. Därefter medelvärdet på de filmer som kunden
hyrt.

-- Först JOINAR jag kund och uthyrning för att ta fram alla kunder som alls
har hyrt en film. Sedan tar jag bort alla de kunder som inte har lämnat
tillbaka ALLA sin filmer. Till sist JOINAR jag de rader jag har kvar i
uthyrningstabellen med filmEx grupperar dem på pNr för att få fram
medelvärdet för var kund som svar.

   SELECT kund.pNr,kund.namn,AVG(filmEx.pris)
   FROM kund,uthyrning,filmEx
   WHERE kund.pNr=uthyrning.pNr AND NOT EXISTS (SELECT * FROM uthyrning
WHERE kund.pNr=uthyrning.pNr AND uthyrning.tillbaka IS NULL) AND
uthyrning.exNr=filmEx.exNr
   GROUP BY kund.pNr;
```

```
-- 16. Hämta telefonnumret för de inkassofirmor som krävt in filmer från
lika många uthyrningar som inkassofirman med telefonnummer 00326573892.
(Tips: använd select count(*) = select count(*)).

-- Först använder jag select count(*) för att ta fram hur många rader som
uppfyller villkoret med rätt telnr. Därefter jämför jag detta mot en ny
select count(*) som först JOINAR inkassofirma och uthyrning och sedan
räknar alla rader som inte har det givna telnr, och om dessa får samma
resultat är villkoret sant.

  SELECT inkassofirma.namn,inkassofirma.tel
  FROM inkassofirma
  WHERE ((SELECT COUNT(*) FROM uthyrning WHERE uthyrning.inkassoTel='00326-
573892')=(SELECT COUNT(*) FROM uthyrning WHERE
inkassofirma.tel=uthyrning.inkassoTel AND uthyrning.inkassoTel!='00326-
573892'));

-- 17. Tanken är att ett filmexemplar inte kan bli uthyrd vid två
tillfällen under samma tidsintervall. Undersök därför om det finns
tidsintervall där ett filmexemplar har blivit uthyrt flera gånger under
samma tidsintervall.

-- Då tipset var att använda sig utav en tempmapp tittade jag först
tillbaka på uppgift 6 och konstaterade att jag måste använda NOT för att
plocka bort jämförelsen mellan identiska rader med hjälp av alla
primärnycklar så jag skrev en NOT(). Därefter ritade jag upp de möjliga
alternativen som skulle resultera i det jag sökte på ett papper. Sedan
skrev jag de 4 olika alternativen i varsin parentes med ett OR emellan som
alltså undersökte varje alternativ jag ville testa och omgav detta tillslut
med en parentes. Till sist skrev jag en AND som bad om de rader där exNr
och ägarbutiken var detsamma så att jag skulle se att det gällde en unik
film som alltså hyrts ut samtidigt till flera.

  SELECT uthyrning.*
  FROM uthyrning, uthyrning temp
  WHERE NOT(uthyrning.utDatum=temp.utDatum AND uthyrning.pNr=temp.pNr AND
uthyrning.exNr=temp.exNr AND uthyrning.agarbutik=temp.agarbutik AND
uthyrning.filmtitel=temp.filmtitel AND
uthyrning.inkassoTel=temp.inkassoTel)
  AND ((temp.utDatum<=uthyrning.utDatum AND
temp.tillbaka>uthyrning.utDatum)
  OR (temp.utDatum<uthyrning.tillbaka AND
temp.tillbaka>=uthyrning.tillbaka)
  OR (temp.utDatum<=uthyrning.utDatum AND
temp.tillbaka>=uthyrning.tillbaka)
  OR (temp.utDatum>=uthyrning.utDatum AND
temp.tillbaka<=uthyrning.tillbaka))
  AND uthyrning.exNr=temp.exNr AND uthyrning.agarbutik=temp.agarbutik;

-- 18. Hämta all information om inkassofirmor som kommer från en stad som
börjar på "E".

-- Villkoret som är att staden skall börja med E tas fram med LIKE och
wildcardet % som lägger till 0-fler tecken efter E. Om någon stad i
tabellen inkassofirma börja med E skrivs alltså all information i den raden
ut.

  SELECT *
  FROM inkassofirma
  WHERE inkassofirma.stad LIKE 'E%';
```

```
242
243 -- 19. Hämta namn och adress för de kunder som angivit ett personnummer som
244 inte är på formen XXXXXXXXXX där X är en siffra mellan 0 och 9. (Tips:
245 använd rlike eller regxp.)
246
247 -- Då jag inte angivit några adresser för mina kunder väljer jag att
248 istället lista tel.
249
250 -- Jag använder mig av regexp() som kontrollerar att strängen börjar(^),
251 innehåller(+) och avslutas($) med en siffra. Sedan använder jag length()
252 för att se till att strängen är exakt 10 tecken. Om villkoret är sant för
253 pNr i tabellen kund, listas namn och tel.
254
255   SELECT kund.namn,kund.tel
256   FROM kund
257   WHERE NOT(kund.pNr REGEXP('^[0-9]+$') AND LENGTH(kund.pNr)=10);
258
259 -- 20. Hämta namn och telefonnummer för den kund som har hyrt den dyraste
260 filmen. (Tips: använd max().)
261
262 -- Jag börjar med att JOINA tabellerna uthyrning,filmEx och kund med
263 varandra. Därefter väljer jag ut de rader som har samma pris som det högsta
264 priset i tabellen filmEx vilket räknas ut med hjälp av max() och som
265 dessutom finns med både i tabellen filmEx och uthyrning..
266
267   SELECT kund.namn,kund.tel
268   FROM kund,uthyrning,filmEx
269   WHERE uthyrning.exNr=filmEx.exNr AND kund.pNr=uthyrning.pNr AND
270 filmEx.pris=(SELECT MAX(filmEx.pris) FROM filmEx,uthyrning WHERE
271 filmEx.exNr=uthyrning.exNr);
272
273 -- 21. Hämta namnet på kunden som senast hyrde filmen Sataythai på Vipp-
274 video.
275
276 -- Jag börjar med att JOINA tabellerna kund och uthyrning. Sedan väljer jag
277 ut de rader där filmtiteln är den efterfrågade och uthyrningsdatumet är det
278 senaste, dvs. det högsta, vilket tas fram med max().
279
280   SELECT kund.namn
281   FROM kund,uthyrning
282   WHERE kund.pNr=uthyrning.pNr AND uthyrning.filmtitel='Sataythai' AND
283 uthyrning.utDatum=(SELECT MAX(uthyrning.utDatum) FROM uthyrning);
284
285 -- 22. Lista de uthyrningar som skett den senaste veckan. (Tips: använd
286 curdate() eller liknande.)
287
288 -- Jag använder date_sub() för att dra av 7 dagar från dagens datum som
289 anges som curdate() och om uthyrningsdatumet är större än datumet för 7 dgr
290 sedan så listas den raden.
291
292   SELECT *
293   FROM uthyrning
294   WHERE uthyrning.utDatum > DATE_SUB(CURDATE(), interval 7 day);
295
296 -- 23. Höj lönen för alla anställda med en månadslön mellan 10000 och 12000
297 kr med 22%.
298
299 -- Jag uppdaterar tabellen anställd och där kolumnen lön har ett värde
300 mellan 10000 och 12000 så sätter jag lönen till att bli 22% högre.
301
302   UPDATE anstalld
```

```
303     SET anstalld.lon=anstalld.lon*1.22
304     WHERE anstalld.lon>=10000 AND anstalld.lon<=12000;
305
306   -- 24. Ta bort exemplaret med nummer 442.
307
308   -- Jag använder kommandot DELETE FROM på exemplartabellen och väljer att ta
309   bort exnr 442. Detta kommer dock inte att fungera eftersom att exNr som i
310   detta fall är en primärnyckel, även är en foreignkey i bl.a. tabellen
311   finns. För att det skall fungera måste alla hänvisningar till det du vill
312   radera, plockas bort.
313
314     DELETE FROM filmEx
315     WHERE filmEx.exNr='442';
316
317   -- 25. Ta bort den anställde med anställningsnummer 31366.
318
319   -- Jag använder kommandot DELETE FROM på anställdtabellen och väljer att ta
320   bort anställningsnummer 31366. Detta kommer dock inte att fungera eftersom
321   att aNr som i detta fall är en primärnyckel, även är en foreignkey i samma
322   tabell då den hänvisar till den chef som den anställde har. För att det
323   skall fungera måste alla hänvisningar till det du vill radera, plockas
324   bort.
325
326     DELETE FROM anstalld
327     WHERE anstalld.aNr='31366';
```