DD1370: Databaser och Informationssystem **Hösten 2014**

Introduktion till frågespråket SQL (v0.9)

Petter Ögren *13:e November*

Disclaimer: *Dessa anteckningar har producerats under viss tidspress, och kan därför innehålla smärre fel och oklarheter. Notera också att dokumentet är avsett att komplettera ppt-kopiorna och inte ersätta dem.*

Sammanfattning Detta dokument beskriver mycket kortfattat de viktigaste delarna av frågespråket SQL (Structured Query Language) för sökning och urval ur databaser. Eftersom en databas består av en eller flera tabeller så kommer det att handla om hur man väljer ut, och kombinerar, data från en eller flera tabeller.

1 SQL-kommandon som beskrivs i dokumentet

För att ge en överblick listar vi här de SQL-kommandon som beskrivs, samt ger ett exempel på hur kommandot används

SELECT Första ordet i all SQL-frågor, används t.ex. på formen `SELECT X FROM Y`

FROM Anger från vilken eller vilka tabeller urvalet görs. Om flera tabeller anges kombineras data ifrån dem ihop på ett sätt som liknar `JOIN`.

WHERE Ett villkor för vilka rader som tas med.

AS Används för att byta namn på kolumner.

LIKE Används för att jämföra teckensträngar. Ofta tillsammans med `%` och `_`.

IS NULL Avgör om data saknas.

ORDER BY Används för att sortera en tabell med avseende på viss kolumn.

MAX, MIN, SUM, AVG, COUNT Används för att göra beräkningar.

GROUP BY Grupperar data för beräkningar på olika delar av det.

HAVING Urval av grupper.

CREATE VIEW AS Skapar en Vy.

IN, NOT IN Avgör om data finns, eller inte finns i en tabell.

JOIN ON Kombinerar flera tabeller.

2 Välja kolumner ur tabeller

I vanliga fall består databaser av ett antal tabeller, där varje tabell består av ett stort antal rader och kolumner. För att få ut just den information man är intresserad av ur en databas använder man kommandot `SELECT`. Vi skall först se hur man kan välja vilka kolumner man vill se.

Den kanske enklaste SQL-frågan av alla är på formen `SELECT X FROM Y`, där `X` är en kolumn och `Y` en tabell. Denna fråga plockar alltså fram kolumnen `X` från tabellen `Y`.

Om vi har följande tabell,

Anställda		
Pnr	Namn	Lön
334455-0000	Kalle	20 000
667788-0000	Anna	30 000
997788-0000	Adam	25 000

och skriver `SELECT "Lön" FROM "Anställda"` så får vi

Lön
20 000
30 000
25 000

Om vi istället skriver `SELECT "Pnr" FROM "Anställda"` så får vi

Pnr
334455-0000
667788-0000
997788-0000

Man kan även räkna upp flera kolumner i samma fråga. Då ser det ut såhär `SELECT X1, X2 FROM Y`.

Till exempel skulle man kunna skriva `SELECT "Namn", "Pnr" FROM "Anställda"`. Vilket ger följande tabell

Namn	Pnr
Kalle	334455-0000
Anna	667788-0000
Adam	997788-0000

Notera att det går utmärkt att skriva kolumnerna i vilken ordning man vill, det behöver inte vara samma ordning som i den ursprungliga tabellen.

Vill man ha alla kolumnerna kan man antingen räkna upp dem. `SELECT "Pnr", "Namn", "Lön" FROM "Anställda"`, eller skriva `SELECT * FROM "Anställda"`. Båda alternativen ger alltså resultatet

Pnr	Namn	Lön
334455-0000	Kalle	20 000
667788-0000	Anna	30 000
997788-0000	Adam	25 000

Vi kan även byta namn på kolumner, genom att använda kommandot `SELECT X1 AS X2 FROM Y`. Nu är `X1` namnet på den kolumn man vill visa, `X2` det nya namn man vill ge den, och `Y` tabellen där kolumnerna finns.

Till exempel kan vi skriva `SELECT "Pnr", "Namn" AS "Lirare", "Lön" FROM "Anställda"`, vilket ger

Pnr	Lirare	Lön
334455-0000	Kalle	20 000
667788-0000	Anna	30 000
997788-0000	Adam	25 000

På detta sätt kan man byta namn på valfritt antal kolumner.

Nu när vi kan välja, och döpa om, de kolumner i tabellen vi vill titta på, så skall vi se hur man väljer vilka rader som skall visas.

3 Välja rader ur tabeller

Det enklaste sättet att välja rader ur en tabell är att använda kommandot `SELECT X1 FROM Y WHERE X2=Z`. Här kan `X1` precis som ovan vara namnet på en eller flera (komma-separerade) kolumner. `X2` är en namnet på den kolumn vi vill använda för urvalet, och `Z` är det värde vi letar efter.

Om vi till exempel skriver `SELECT "Namn", "Lön" FROM "Anställda" WHERE "Lön"= 25 000` så får vi följande

Namn	Lön
Adam	25 000

Här kan vi också använda de matematiska symbolerna `<` (mindre än) `<=` (mindre än eller lika med) `>` (större än) `>=` (större än eller lika med) och `<>` (inte lika med).

Om vi till exempel skriver `SELECT "Namn", "Lön" FROM "Anställda" WHERE "Lön">= 25 000`, så får vi

Namn	Lön
Anna	30 000
Adam	25 000

Om vi vill göra urvalet baserat på text-information (t.ex. namn) istället för siffror (t.ex. lön) så använder vi istället `SELECT X1 FROM Y WHERE X2 LIKE Z`. Precis som ovan är `X1` namnet på en eller flera (komma-separerade) kolumner, `X2` är en namnet på den kolumn vi vill använda för urvalet, och `Z` är det värde vi letar efter. För att ange `Z` använder vi dock symbolerna `%` (godtycklig textsträng) och `_` (godtycklig bokstav).

Om vi till exempel skriver `SELECT * FROM "Anställda" WHERE "Namn" LIKE 'A%'`, så får vi all information om alla vars namn börjar på A:

Pnr	Namn	Lön
667788-0000	Anna	30 000
997788-0000	Adam	25 000

Om vi skriver `SELECT * FROM "Anställda" WHERE "Namn" LIKE 'An%'` så får vi alla vars namn börjar på An.

Pnr	Namn	Lön
667788-0000	Anna	30 000

Skriver vi `SELECT * FROM "Anställda" WHERE "Namn" LIKE '%a%'` så får vi alla vars namn har ett litet a någonstans (den godtyckliga textsträngen `%` står på båda sidorna om a-et, alltså är det ok med andra bokstäver, 0 eller flera, på båda sidor om ett a, så länge det finns något a med).

Pnr	Namn	Lön
334455-0000	Kalle	20 000
667788-0000	Anna	30 000
997788-0000	Adam	25 000

Symbolen `_` fungerar ungefär som `%` men den motsvarar exakt en bokstav (vilken som helst), inte en sträng av bokstäver.

Om vi skriver `SELECT * FROM "Anställda" WHERE "Namn" LIKE '_____'` så får vi alla namn som är exakt 5 bokstäver långa

Pnr	Namn	Lön
334455-0000	Kalle	20 000

och skriver vi `SELECT * FROM "Anställda" WHERE "Namn" LIKE 'A__a'` så får vi alla namn som är fyra bokstäver långa, börjar med A och slutar med a

Pnr	Namn	Lön
667788-0000	Anna	30 000

Vi kan även kombinera de två, `SELECT * FROM "Anställda" WHERE "Namn" LIKE '%a_'` ger alla vars namn har a som näst sista bokstav

Pnr	Namn	Lön
997788-0000	Adam	25 000

Slutligen finns en symbol för fält som inte är ifyllda. Skriver vi

```
SELECT * FROM "Anställda" WHERE "Lön" is Null
```

Får vi ut alla anställda som inte har ett lönebelopp inlagt i databasen. I detta fall blir det en tom tabell, eftersom alla fält är ifyllda i vårt exempel.

4 Sortering, Funktioner och beräkningar

Databasers främsta uppgift är att hantera och kombinera data. Man kan även göra vissa beräkningar i SQL, men för komplicerade beräkningar är det ofta bättre att flytta över datan till ett kalkylbladsprogram (t.ex. Excel) eller använda ett generellt programmeringsspråk (t.ex. Java).

För att sortera en tabell använder vi kommandot `SELECT X1 FROM Y ORDER BY X2`. Här görs precis som tidigare ett urval, men den resulterande tabellen sorteras dessutom, utifrån värdena i kolumnen `X2`. Vill man

dessutom bestämma om sorteringen skall vara stigande (ascending) eller fallande (descending) så lägger man till ett ASC eller DESC på slutet.

Om vi skriver `SELECT * FROM "Anställda" ORDER BY "Namn"` så får vi

Pnr	Namn	Lön
997788-0000	Adam	25 000
667788-0000	Anna	30 000
334455-0000	Kalle	20 000

Notera att om vi inte skriver ut ASC eller DESC så blir det automatiskt (default) ASC.

Skriver vi istället `SELECT * FROM "Anställda" ORDER BY "Lön"` så får vi

Pnr	Namn	Lön
334455-0000	Kalle	20 000
997788-0000	Adam	25 000
667788-0000	Anna	30 000

och om vi skriver `SELECT * FROM "Anställda" ORDER BY "Lön" DESC` så får vi

Pnr	Namn	Lön
667788-0000	Anna	30 000
997788-0000	Adam	25 000
334455-0000	Kalle	20 000

Vi kan även göra beräkningar på utvalda kolumner. Då använder vi funktionerna `max` (största värdet), `min` (minsta värdet), `avg` (medelvärdet), `sum` (summan), `count` (antalet värden).

Om vi till exempel vill hitta den lägsta lönen i tabellen skriver vi `SELECT MIN("Lön") FROM "Anställda"`, vilket ger

MIN(Lön)
20 000

På liknande sätt kan vi skriva `SELECT MAX("Lön") FROM "Anställda"`

MAX(Lön)
30 000

`SELECT AVG("Lön") FROM "Anställda"` ger

AVG(Lön)
25 000

`SELECT SUM("Lön") FROM "Anställda"` ger

SUM(Lön)
75 000

och `SELECT COUNT("Lön") FROM "Anställda"` ger

COUNT(Lön)
3

Notera att `count` räknar antalet ifyllda fällt, så om ett fält är tomt (kan kollas med `is NULL` enligt ovan) så kommer det inte med.

Vi kan också använda de fyra räknesätten på kolumner med siffror.

Till exempel ger
`SELECT "Pnr", "Namn", "Lön", "Lön"*"Lön"/1000000 AS "Kvadratlön mkr" FROM "Anställda"` följande

Pnr	Namn	Lön	Kvadratlön mkr
997788-0000	Adam	25 000	625
667788-0000	Anna	30 000	900
334455-0000	Kalle	20 000	400

Man kan även göra beräkningar över delgrupper av data. Till exempel räkna ut medellöner för män respektive kvinnor. Antag att vi har en lite större tabell.

Anställda			
Pnr	Namn	Kön	Lön
331111-0000	Kalle	Man	20 000
661111-0000	Anna	Kvinna	30 000
991111-0000	Adam	Man	25 000
771111-0000	Amanda	Kvinna	40 000

Då kan vi antingen göra två separata beräkningar.

Först skriver vi `SELECT AVG("Lön") FROM "Anställda" WHERE "Kön"='Kvinna'`, vilket ger

AVG(Lön)
35 000

och sedan skriver vi `SELECT AVG("Lön") FROM "Anställda" WHERE "Kön"='Man'`.

AVG(Lön)
35 000

Man kan även få resultatet samlat i en beräkning genom att använda `GROUP BY`.

Då skriver man `SELECT AVG("Lön") FROM "Anställda" GROUP BY "Kön"` vilket ger

AVG(Lön)
22 500
35 000

man kan även skriva `SELECT "Kön", AVG("Lön") FROM "Anställda" GROUP BY "Kön"`

Kön	AVG(Lön)
Man	22 500
Kvinna	35 000

så blir det tydligare vad som hör till vad. Formeln för `GROUP BY` skulle alltså kunna skrivas
`SELECT X1, fcn(X2) FROM Y GROUP BY X1`, där `X1` är den kolumn man vill samla data med avseende på, och `X2` den kolumn man vill göra beräkningen med. `fcn` skall representera någon funktion, t.ex. `MIN`, `AVG` eller `SUM`. `Y` är som vanligt tabellen där kolumnerna finns.

Låt oss slutligen titta på ett ännu större exempel, där vi använder både `GROUP BY`, `WHERE`, och något som heter `HAVING`.

Anställda			
Pnr	Namn	Kön	Lön
331111-0000	Kalle	Han	20 000
661111-0000	Anna	Hon	30 000
991111-0000	Adam	Han	25 000
771111-0000	Amanda	Hon	40 000
881111-0000	Kim	Hen	40 000

Om vi nu skriver `SELECT "Kön", AVG("Lön") FROM "Anställda" GROUP BY "Kön"` så får vi

Kön	AVG(Lön)
Han	22 500
Hon	35 000
Hen	40 000

Genom att lägga till `HAVING` kan vi göra ett urval över grupperna som `GROUP By` har skapat.
`SELECT "Kön", AVG("Lön") FROM "Anställda" GROUP BY "Kön" HAVING AVG("Lön") > 23 000`

Kön	AVG(Lön)
Hon	35 000
Hen	40 000

Vi kan även använda `WHERE` precis som vanligt. Notera att skillnaden är att `WHERE` gör ett urval innan grupperingen med tillhörande beräkning, och `HAVING` gör ett urval efter grupperingen, på resultatet.

I exemplet kan vi skriva
```
SELECT "Kön", AVG("Lön") FROM "Anställda" WHERE "Lön" > 20000
GROUP BY "Kön" HAVING AVG("Lön") > 23 000
```

Kön	AVG(Lön)
Han	25 000
Hon	35 000
Hen	40 000

Detta gör nu att Adam filtreras bort redan innan snittlönberäkningen, vilket i sin tur gör att snittlönen för männen blir högre, 25 000, och inte tas bort från sluttabellen.

5 Spara sökning i vyer

I detta avsnitt skall vi se hur man kan spara sökningar i så kallade vyer (view). En vy är alltså en SQL-fråga som fått ett eget namn, så att man sedan kan behandla den som vilken annan tabell som helst när man gör nya sökningar. En viktig skillnad från vanliga tabeller är dock att vyn får sin data från andra tabeller, och uppdateras så fort dessa uppdateras.

Ytterligare en viktig skillnad är att vyer behandlas olika i standard SQL och i mjukvaran Open Office Base. Vi skall först beskriva standard SQL och sedan Base.

I standard SQL skapar man en vy genom kommandot `CREATE VIEW Y1 AS S1`, här är `Y1` det nya namnet man vill ge vyn och `S1` är ett helt vanligt `SELECT`-kommando. Vilket som helst av alla de vi sett ovan funkar bra. Om vi till exempel har tabellen

Anställda			
Pnr	Namn	Kön	Lön
331111-0000	Kalle	Han	20 000
661111-0000	Anna	Hon	30 000
991111-0000	Adam	Han	25 000
771111-0000	Amanda	Hon	40 000
881111-0000	Kim	Hen	40 000

så kan vi skriva `CREATE VIEW "Anställda kvinnor" AS SELECT * FROM "Anställda" WHERE "Kön"='Hon'` resultatet blir då

Anställda kvinnor			
Pnr	Namn	Kön	Lön
661111-0000	Anna	Hon	30 000
771111-0000	Amanda	Hon	40 000

I Base funkar det på samma sätt, men skillnaden är att man aldrig skriver `CREATE VIEW Y1 AS S1`. Istället går man in på ett annat ställe i interfacet, och skriver sin fråga direkt. Man går in under Tabeller/Skapa vy". Vyn dyker då också upp bredvid de andra tabellerna. Detta understryker att en Vy kan användas på precis samma sätt som en befintlig tabell i det fortsatta arbetet.

6 Sökning från flera tabeller

Ofta behöver man kombinera information från flera olika tabeller. I SQL finns flera sätt att göra detta. Vi skall titta på så kallade nästlade sökningar, samt två sätt att direkt kombinera olika tabeller.

6.1 Nästlade frågor

En nästlad fråga är en fråga där tabellen man söker ur också är resultatet av en fråga. Kommandot ser ut på följande sätt:

```
SELECT X2 FROM Y2 WHERE X3 IN (SELECT X1 FROM Y1)
```

Här ovan är den inre frågan `(SELECT X1 FROM Y1)` alltså insmugen (nästlad) i den yttre frågan. Resultatet av `(SELECT X1 FROM Y1)` är (som alltid) en tabell, och det är ur den tabellen den yttre frågan hämtar sin data.

Vi har följande tabeller i vår databas.

Anställda			
Pnr	Namn	Kön	Lön
331111-0000	Kalle	Han	20 000
661111-0000	Anna	Hon	30 000
991111-0000	Adam	Han	25 000
771111-0000	Amanda	Hon	40 000
881111-0000	Kim	Hen	40 000

Datorer		
InvNr	Beskrivning	Användare
1	DellPC	331111-0000
2	Macbook Pro	661111-0000
3	HP stationär	881111-0000
4	Dell Laptop	881111-0000

Skall vi ta reda på vilken dator Kalle har kan vi förstås manuellt först leta rätt på Kalle i tabellen Anställda. Skriva upp hans personnummer. Sedan leta reda på personnumret i kolumnen Användare i tabellen Datorer.

I SQL motsvaras detta av följande två frågor:

`SELECT "Pnr" FROM "Anställda" WHERE "Namn"='Kalle'`, vilket ger

Pnr
331111-0000

Sedan skriver vi `SELECT "Beskrivning" FROM "Datorer" WHERE "Användare"='331111-0000'`, vilket ger `SELECT "Pnr" FROM "Anställda" WHERE "Namn"='Kalle'`, vilket ger

Beskrivning
DellPC

Detta förutsätter dock att vi manuellt håller reda på 331111-0000 genom att t.ex. skriva upp det på en lapp. Vill man undvika detta kan man använda en Nästlad fråga.

```
SELECT "Beskrivning" FROM "Datorer" WHERE "Användare" IN
(SELECT "Pnr" FROM "Anställda" WHERE "Namn"='Kalle')
```

Vilket direkt ger

Beskrivning
DellPC

Notera att vi använder `IN` istället för `=`. Anledningen till detta är att vi (i allmänhet) inte vet om den inre frågan ger ett enda svar, eller en hel tabell. Om det finns flera Kalle i tabellen Anställda kommer vi ju att få ut en tabell med alla personnummer till personer som heter Kalle. Skriver vi då `=` så får vi ett felmeddelande (Base kan inte jämföra ett personnummer med en hel tabell av personnummer). Skriver vi däremot `IN` så kollar Base om personnumret finns i tabellen. Och råkar tabellen bestå av bara ett nummer (om det bara finns en Kalle) så funkar det lika bra det också.

Ett exempel på detta får vi om vi istället skriver

```
SELECT "Beskrivning" FROM "Datorer" WHERE "Användare" IN
(SELECT "Pnr" FROM "Anställda" WHERE "Namn" LIKE 'K%')
```

Detta ger

Beskrivning
DellPC
HP Stationär
Dell Laptop

Anledningen är att den inre frågan `(SELECT "Pnr" FROM "Anställda" WHERE "Namn" LIKE 'K%')` ger en tabell med både Kalle och Kim, och den yttre frågan sedan returnerar datorer som tillhör dessa båda. Notera att vi använde `LIKE 'K%'` istället för `='Kalle'` för att få med både Kalle och Kim.

Man kan även leta efter motsatsen till `IN`, dvs allt som inte finns med i tabellen. Då skriver man `NOT IN`. Gör vi denna ändring:

```
SELECT "Beskrivning" FROM "Datorer" WHERE "Användare" NOT IN
(SELECT "Pnr" FROM "Anställda" WHERE "Namn" LIKE 'K%')
```

så får vi alla datorer som inte tillhör Kalle eller Kim. Svaret blir

Beskrivning
Macbook Pro

6.2 Join-frågor

Join-frågor används då man vill 'klistra ihop' data från flera tabeller. Kommandot ser ut på följande sätt `SELECT X3 FROM Y1 JOIN Y2 ON X1=X2`, där `X3` är det man vill se av den kombinerade tabellen (notera att eftersom frågar innefattar flera tabeller måste man ange både kolumn och tabell om inte kolumnnamnen är unika, se nedan), `Y1` och `Y2` är två tabeller, `X1` och `X2` är de kolumner som skall vara lika i 'ihopklistringen' (se nedan). Andra villkor än likhet kan också användas (se nedan).

Om vi har tabellerna Anställda och Datorer ovan, och återigen vill ha reda på vilken dator som Kalle har kan vi nu skriva
`SELECT "Namn", "Beskrivning" FROM "Datorer" JOIN "Anställda" ON "Användare"="Pnr"`, vilket ger

Namn	Beskrivning
Kalle	DellPC
Anna	HP Stationär
Kim	Dell Laptop
Kim	Macbook Pro

Vi kan nu notera att varken Adam eller Amanda kom med i svaret, eftersom de inte har några datorer i listan. Dessutom kom Kim med två gånger, eftersom hen har två datorer.

Vill vi ha med bara raden med Kalle kan vi lägga till ett where-villkor i slutet.

```
SELECT "Namn", "Beskrivning" FROM "Datorer" JOIN "Anställda" ON "Användare"="Pnr"
WHERE "Namn"='Kalle'
```
, vilket ger

Namn	Beskrivning
Kalle	DellPC

Innan vi beskriver nästa sätt att kombinera tabeller skall vi notera två saker.

För det första kan det ibland vara så att kolumnnamnen inte är unika om man jobbar med flera tabeller samtidigt. För att vara på säkra sidan skriver man då `"tabell"."kolumn"` istället för bara `"kolumn"`. I exemplet ovan blir det då

```
SELECT "Anställda"."Namn", "Datorer"."Beskrivning"
FROM "Datorer" JOIN "Anställda" ON "Datorer"."Användare"="Anställda"."Pnr"
WHERE "Anställda"."Namn"='Kalle'
```

och resultatet blir precis detsamma.

Det andra vi skall notera är att man kan ha andra sorters jämförelser än = för 'ihopklistringen'. T.ex. kan vi ersätta likheten med en olikhet (vi lägger även till en sortering på namn för tydlighetens skull):

```
SELECT "Namn", "Beskrivning" FROM "Datorer" JOIN "Anställda" ON "Användare"<"Pnr"
ORDER BY "Namn"
```

Detta ger följande resultat

Namn	Beskrivning
Adam	Dell Laptop
Adam	HP Stationär
Adam	Macbook Pro
Adam	DellPC
Amanda	Macbook Pro
Amanda	DellPC
Anna	DellPC
Kim	DellPC
Kim	Macbook Pro

Tabellen visar för varje anställd vilka datorer anställda som är äldre (lägre Pnr) än hen har. Kalle är äldst, och det finns inga äldre anställda med datorer, vilket gör att han inte kommer med i listan. Adam är å andra sidan yngst, och har ingen egen dator, så alla fyra datorerna ägs av äldre personer, och kommer med vid hans namn.

6.3 Select från flera tabeller

Ett alternativt sätt att kombinera tabeller är en vanlig select-fråga, där man räknar upp flera tabeller. Det ser ut enligt följande:
`SELECT X3 FROM Y1, Y2 WHERE X1=X2`,
där `X3, X2, X1, Y1, Y2` är precis som i Join-frågan ovan.

Skriver vi alltså:
`SELECT "Namn", "Beskrivning" FROM "Datorer", "Anställda" WHERE "Användare" = "Pnr"`, så får vi samma resultat som
`SELECT "Namn", "Beskrivning" FROM "Datorer" JOIN "Anställda" ON "Användare" = "Pnr"`, det vill säga

Namn	Beskrivning
Kalle	DellPC
Anna	HP Stationär
Kim	Dell Laptop
Kim	Macbook Pro

Vad händer då om vi skriver `SELECT X FROM Y1, Y2`? Eftersom vi inte har något villkor på hur 'ihopklistringen' går till så kommer tabellerna att kombineras på precis alla sätt som är möjligt.

Betrakta följande lite mindre tabeller:

Höginkomsttagare			
Pnr	Namn	Kön	Lön
661111-0000	Anna	Hon	30 000
771111-0000	Amanda	Hon	40 000
881111-0000	Kim	Hen	40 000

Dyra datorer		
InvNr	Beskrivning	Användare
2	Macbook Pro	661111-0000
3	HP stationär	881111-0000
4	Dell Laptop	881111-0000

Skriver vi nu `SELECT * FROM "Höginkomsttagare", "Dyra datorer"` så får vi

Pnr	Namn	Kön	Lön	InvNr	Beskrivning	Användare
661111-0000	Anna	Hon	30 000	2	Macbook Pro	661111-0000
661111-0000	Anna	Hon	30 000	3	HP stationär	881111-0000
661111-0000	Anna	Hon	30 000	4	Dell Laptop	881111-0000
771111-0000	Amanda	Hon	40 000	2	Macbook Pro	661111-0000
771111-0000	Amanda	Hon	40 000	3	HP stationär	881111-0000
771111-0000	Amanda	Hon	40 000	4	Dell Laptop	881111-0000
881111-0000	Kim	Hen	40 000	2	Macbook Pro	661111-0000
881111-0000	Kim	Hen	40 000	3	HP stationär	881111-0000
881111-0000	Kim	Hen	40 000	4	Dell Laptop	881111-0000

Noter att data från samma tabell alltid hänger ihop som tidigare, exempelvis har Anna alltid 30000 i lön, och Macbook Pro alltid InvNr 2. Rader från olika tabeller har dock kombinerats på all 9 sätt som går, utgående från 3 anställda och 3 datorer ($3 \times 3 = 9$).

Om 'ihopklistringen' skall ha någon mening måste vi dock i detta fall kräva att Pnr från Anställdatabellen stämmer med Användare från Datortabellen. Då skriver vi
`SELECT * FROM "Höginkomsttagare", "Dyra datorer" WHERE "Pnr"="Användare"`

och då får vi

Pnr	Namn	Kön	Lön	InvNr	Beskrivning	Användare
661111-0000	Anna	Hon	30 000	2	Macbook Pro	661111-0000
881111-0000	Kim	Hen	40 000	3	HP stationär	881111-0000
881111-0000	Kim	Hen	40 000	4	Dell Laptop	881111-0000

Vill vi sedan bara ha med Namn och Beskrivning i tabellen byter vi ut * enligt följande
`SELECT "Namn", "Beskrivning" FROM "Höginkomsttagare", "Dyra datorer" WHERE "Pnr"="Användare"`
vilket ger.

Namn	Beskrivning
Anna	HP Stationär
Kim	Dell Laptop
Kim	Macbook Pro

Notera att anledningen till att Kalle inte är med i resultatet är att han inte är med Höginkomsttagare-tabellen (och hans dator inte är med i Dyra datorer).