SQL

SQL, standard query language, är ett språk för att tala med en databas och tabellerna i databasen. Trots dess namn finns det en mängd olika dialekter av SQL, som exempelvis Microsoft SQL Server, MySQL och PostgreSQL för att nämna några. Grundfunktionerna är dock desamma – och variationerna brukar vara lätta att hitta på nätet. Det svåra är egentligen inte språket, utan hur man använder det – och hur man ska tänka.

Här kommer vi att använda PostgreSQL som exempel. Det är open source och kommer dessutom med ett enkelt användargränssnitt som gör det lite enklare att arbeta i. Man slipper skriva sina kommandon i kommandofönstret.

**Skapa en databas**

En databas kan innehålla flera tabeller. För lokala databaser brukar jag behålla postgres som ägare. Se också till att du ändrar kodningen under rubriken Definition – klicka på Collation och välj Swedish (eller Norwegian) och gör samma sak för Character Type.

Övriga flikar behöver inte anges när man jobbar lokalt för sig själv. Klicka OK för att skapa databasen.

**Skapa tabeller**

Till skillnad från ex Access så måste man skapa tabellen innan man fyller den med innehåll. Det gäller att veta vilka datatyper man vill ha och hur långa fälten ska vara. Det går att ändra konfigurationen på tabellen efter att man skapat den, men det lönar sig att fundera lite innan man går igång. Tänk på att till skillnad från Excel så är en databas mycket mer kinkig med detaljerna. Det går inte att läsa in ett textvärde till en numerisk datatyp till exempel. Om du har ett enda värde som är längre än den längd på fältet som du angivit så kommer hela inläsningen att förkastas.

**Datatyper**

Numeriska: **integer** (går inte att ha hur stora värden som helst, gräns ±2,1 miljarder)

**double precision** (måste ha punkt istället för komma för att markera decimaler)

Text: **character varying(n)**, text (för riktigt långa texter)

Tid: **date, time**

***ÖVNING: Skapa en tabell över folk i rummet***

CREATE TABLE personer

(

 fornamn character varying(128),

 efternamn character varying(128),

 foretag character varying(128),

 land character varying(128),

 alder integer

);

**Mata in personer**

INSERT INTO personer (fornamn, efternamn, foretag, land, alder) VALUES (’Helena’, ’Bengtsson’, ’SVT’, ’Sverige’, 46);

Tänk på att du måste ha raka enkla citattecken. Det lönar sig att använda en editor som exempelvis UltraEdit eller Notepad ++ om man tänker jobba mycket med det här. För enklare övningar passar Anteckningar i Windows alldeles utmärkt.

**Mata in flera**

Om man behöver mata in ett gäng poster så är Excel – eller ett annat spreadsheet som Google Docs eller Numbers ett utmärkt verktyg. Med största sannolikhet så har du redan din data där – och då är det väldigt enkelt att med hjälp av textformler skapa ett antal INSERT-kommandon. PostgreSQL tillåter också att du kör ett stort antal kommandon på en gång, så länge som du inte har något fel i dem.

Exempel på formel i Excel:

=”INSERT INTO personer (fornamn, efternamn, foretag, land, alder) VALUES (’”&A2&”’, ’”&B2&”’, ’”&C2&”’, ’”&D2&”, ”&E2&”);”

Kontrollera kommandot innan du kopierar det för de övriga raderna

**Urval**

SELECT \* FROM personer;

SELECT fornamn, alder FROM personer;

**Villkor**

SELECT \* FROM personer WHERE land=’Sverige’;

SELECT fornamn, efternamn, foretag FROM personer WHERE foretag ilike ’%aft%’;

SELECT \* FROM personer WHERE foretag in (’SVT’, ’SVD’, ’Aftonbladet’);

SELECT \* FROM personer WHERE alder < 35;

**Sortera**

SELECT \* FROM personer ORDER BY alder;

SELECT \* FROM personer WHERE land=’Norge’ ORDER BY foretag;

**Gruppera och räkna**

SELECT land, COUNT(fornamn) FROM personer GROUP BY land;

SELECT land, foretag, COUNT(\*) FROM personer GROUP BY land, foretag ORDER BY 3 DESC;

**Funktioner**

På samma sätt som i Excel så finns det mängder av funktioner i PostgreSQL. Man kan använda char\_length(efternamn) för att se längden på ett fält eller substring(foretag from 1 for 4) för att hämta de fyra första tecknen i ett fält. Se ex <http://www.postgresql.org/docs/9.1/static/functions-string.html> för mer funktioner eller sök på postgres och den funktion du vill använda.

**Ändra layout på tabell**

ALTER TABLE personer ADD COLUMN civilstand character varying(6);

ALTER TABLE personer ADD COLUMN idnr serial NOT NULL;

(skapar en räknare och unikt idnr för varje person)

**Uppdatera tabeller**

UPDATE personer SET civilstand=’ogift’ where idnr=1;

På samma sätt som med INSERT så kan man använda Excel för att bygga ihop dessa kommandon. Var dock noga med att ta med villkoret – och se till att det är unikt. Kör du uppdateringskommandot utan villkor kan det innebära att du byter namn på hela databasen. Därför är det lämpligt att skapa unika id:n för varje post, det blir lättare att arbeta med tabellen och sökningen går också snabbare.

Ett tips är att först köra frågan som en SELECT – då ser man hur många rader som påverkas och det kan kännas tryggare att sedan köra uppdateringen.

**Ta bort rader**

Ett än mer förödande kommando är det som tar bort rader. Glömmer man villkoret här så tömmer man tabellen. Och till skillnad från Access så frågar inte PostgreSQL innan och talar om hur många rader som tas bort.

DELETE FROM personer where foretag=’VG’;

Även här brukar jag köra ett SELECT kommando först och sedan byta ut texten till DELETE för att vara säker på att jag inte tar bort fel, eller mer än vad jag vill.

**Inläsning och samkörning**

Det vanliga är ju att man läser in data i form av text-filer till tabeller som man sedan fortsätter att jobba med. Som tidigare nämnt måste man börja med att skapa tabellerna innan man läser in data. Vi ska jobba med två tabeller med data från Skolverket:

CREATE TABLE elever

(skola character varying(128),

antal\_elever\_f integer,

andel\_flickor\_f integer,

antal\_elever integer,

andel\_flickor integer,

andel\_utl integer,

andel\_utb integer,

antal\_elever\_1 integer,

antal\_elever\_2 integer,

antal\_elever\_3 integer,

antal\_elever\_4 integer,

antal\_elever\_5 integer,

antal\_elever\_6 integer,

antal\_elever\_7 integer,

antal\_elever\_8 integer,

antal\_elever\_9 integer);

CREATE TABLE personal

(

skola character varying(255),

antal\_personal double precision ,

antal\_ledare double precision ,

antal\_larare double precision ,

andel\_kvinnor double precision ,

andel\_pedex double precision ,

andel\_spec double precision ,

elever\_larare double precision ,

antal\_syv double precision ,

elever\_perSYV double precision );

**Förbereda filer för inmatning till SQL**

Filen måste vara i kodformatet UTF-8 för att det ska funka. UTF-8 är en språkkod som gör att Å, Ä och Ö visas på rätt sätt.

Om du har ditt material i Excel så brukar jag först spara filen i SKV-format – semikolon-separerad – och sedan öppna filen i Anteckningar, eller Notepad++ eller UltraEdit och spara den som UTF-8. Om du har decimaltal måste du byta ut kommat mot en punkt, ex 4,5 blir 4.5. Använd Excel:s Sök och Ersätt för att göra detta.

För PostgreSQL använder man sedan kommandot COPY för att ladda upp filen.

Ett kommando kan se ut såhär:

COPY personal (skola, antal\_personal, antal\_ledare, antal\_larare,

andel\_kvinnor, andel\_pedex, andel\_spec, elever\_larare, antal\_syv, elever\_per\_syv) FROM 'C:/DB/personal.txt' WITH DELIMITER AS ';' CSV HEADER;

Man kan också använda COPY för att hämta ut data – men normalt gör jag Ctrl-C, Ctrl-V till Excel efter att jag har sökt ut den data jag vill ha. Då hamnar alla fält i kolumn A i Excel. Därefter använder jag Data – Text till kolumn för att placera fälten i var sin kolumn. Tänk på att ändra format på de kolumner som ska vara text, som exempelvis läns- och kommunkoder. Ett knep är att först markera kolumnen och välja Formatera celler och sedan Text, innan du kopierar in från PostgreSQL.

**Ihopkoppling**

Att koppla ihop två, eller flera, tabeller är ju egentligen huvudsyftet med att använda en relationsdatabas. Det finns två sätt att koppla ihop, inner join och outer join. Man kan dels använda dessa i ett urvalskommando – för att veta vilka poster som är lika i två tabeller – dels kan man använda join-kommandot för att uppdatera en tabell med en annan.

Om man har döpt något fält till samma namn i de två tabellerna måste man inkludera tabellnamnet när man hänvisar till fältet genom att skriva tabell.falt

SELECT elever.skola, antal\_elever, antal\_larare FROM elever INNER JOIN personal ON elever.skola=personal.skola WHERE antal\_elever > 100;

En outer join betyder att man tar med alla fält i en tabell – och de fält i den andra tabellen som matchar den första. Detta kan också användas för att hitta de fält som finns i tabell a men inte i b eller tvärtom.

Hitta skolor som finns i elevtabellen men inte i personaltabellen:

SELECT elever.skola, personal.skola, antal\_elever, antal\_larare FROM elever LEFT OUTER JOIN personal ON elever.skola=personal.skola WHERE personal.skola IS NULL;

Hitta skolor som finns i personaltabellen men inte i elevtabellen:

SELECT elever.skola, personal.skola, antal\_elever, antal\_larare FROM elever RIGHT OUTER JOIN personal ON elever.skola=personal.skola WHERE elever.skola IS NULL;

Uppdatera med hjälp av en annan tabell

UPDATE personal SET antal\_elever=elever.antal\_elever FROM elever WHERE personal.skola=elever.skola;

**Back up**

För att göra back up måste man använda kommandoprompten. Det finns en mängd olika inställningar som kan göras – här är exempel på några:

pg\_dump -h 127.0.0.1 --no-owner -f c:/db/inkomst/inkomstdata.sql -U postgres inkomst – dumpar alla tabeller från databasen inkomst

pg\_dump -h 127.0.0.1 -U valpejl --data-only -f C:/DB/TillDB/ personinfo.sql -t person\_info valpejl – dumpar bara innehållet från tabellen person\_info (inte datastrukturen) från databasen valpejl till filen person\_info.sql

**Ytterligare hjälp**

Det finns massvis med information och forum på nätet som handlar om olika SQL-problem och hur man löser dem. Enklast att hitta är faktiskt att beskriva problemet på google och lägga till postgres för att vara säker på att man får svaret i den dialekt av SQL som postgreSQL använder.

PostgreSQL har en ganska bra och väldigt omfattande manual på nätet - [http://www.postgresql.org](http://www.postgresql.org/) och den är väl indexerad på google så att det är lätt att hitta dit. Sök exempelvis på ***postgres string functions*** för att få en tabell över alla funktioner som finns i postgres.