SQL

Structured Query Language

Ett sk. Frågespråk.

Skapades på 70 talet för att hantera data i relationsdatamodeller.

SQL

Har man ingen egen databashanterare installerad så går det att använda nåt online verktyg med tillhörande test databas.

Exempelvis:

https://www.sqltutorial.org/seeit/

Eller:

https://www.w3schools.com/sql/default.asp

SQL

SQL består egentligen av tre ’språk’, dvs det används på tre olika sätt:

- **Data Definition language** används när vi vill skapa databaser och tabeller etc(CREATE TABLE, CREATE DATABASE etc)
- **Data Management Language** används när vi vill använda data i våra databaser och tabeller (SELECT, INSERT, UPDATE etc)
- **Data Control Language** används för att kontrollera behörighet mm på vem som får använda datan (GRANT, REVOKE etc)

Skapa databas

För att kunna lägga in nåt i en databas så behöver ju först en databas att lägga in nåt i.

Det gör man med kommandot

CREATE DATABASE namn_på_databas;

(obs, detta funkar inte på exempelvis SQLite baserade verktyg)

Välja databas

När man skapat en ny databas så behöver man tala om för sin databashanterare att man vill använda just den databasen. Det gör man med **USE**.

USE namn_på_databas;

(ej SQLite)

Skapa nya tabeller

När man väl har en databas så behöver man bestämma vad som ska finnas i databasen.

Detta var ju en av nackdelarna med relationsdatabaser: att man behöver strukturera upp datan innan man kan använda den. Syntaxen för att skapa en ny tabell är:

CREATE TABLE namn_på_tabell (kolumn1 datatyp, kolumn2 datatyp, kolumn3 datatyp…);

Namnkonventionen är att man anger tabellnamn i plural (persons, cars, animals etc)

Strings i SQL

Många datyper är samma i SQL som i java/kotlin. int, double mm.

Däremot finns inte String i SQL utan istället använder man **CHAR(size), VARCHAR(max_size) och TEXT.**

Ex:

```
name CHAR(20)
adress VARCHAR(255)
email TEXT
```

Man anger i de två första mao hur lång ’strängen’ ska vara.

Skillnaden mellan dessa är att VARCHAR varierar längden upp till max det angivna värdet, medan CHAR alltid har den angivna länden (är strängen kortare så fylls det på med extra blanksteg upp till max)

TEXT är dynamisk och anpassar längden efter innehållet utan något max tak

Ta bort tabeller

Ibland vill man ta bort en tabell (framför allt medan man experimenterar)

Det gör man med kommandot DROP namn_på_tabell.

Var försiktiga dock då all data som fanns i tabellen försvinner om vi tar bort tabellen

Ändra tabeller

Man kan även lägga till eller ta bort kolumner i befintliga tabeller, det görs med kommandot

ALTER TABLE tabell_namn

följt av operationen man vill utföra.

ALTER TABLE tabell_namn ADD kolumn_namn datatyp;

ALTER TABLE tabell_namn DROP kulumn_namn;

Tar man bort en kolumn så försvinner all data som fanns lagrad i den

Constraints

När man skapar kolumner (antingen med create table eller alter table) så kan man även ange **Constraints** till dessa. Dvs vilkor för vad just den kolumnen ska innehålla. Några vanliga constraints:

- `NOT NULL` - Ensures that a column cannot have a NULL value
- `UNIQUE` - Ensures that all values in a column are different
- `PRIMARY KEY` - A combination of a `NOT NULL` and `UNIQUE`. Uniquely identifies each row in a table
- `FOREIGN KEY` - Prevents actions that would destroy links between tables
- `CHECK` - Ensures that the values in a column satisfies a specific condition
- `DEFAULT` - Sets a default value for a column if no value is specified
- `CREATE INDEX` - Used to create and retrieve data from the database very quickly

Primary Key

Primary Key är ju standard parametern man använder för att komma åt en specifik rad, därför har en Primary Key constraint automatiskt NOT NULL och UNIQUE constraints.

```
CREATE TABLE persons (
        Id int PRIMARYKEY,
        Name varchar(50),
        Email varchar(50)
);
```

Foreign Key

Ett syfte med relationsdatabaser är att man vill koppla ihop tabeller med varandra. Foreign Key är en referens till en Primary Key i en annan tabell.

```
CREATE TABLE cars(
        id int PRIMARY KEY,
        model varchar(50),
        owner int,
        FOREIGN KEY (owner) REFERENCES persons(id)
);
```

Foreign Key

Kopplar man ihop tabeller med Foreign-Primary keys så blir tabellen med Foreign Key beroende av Tabellen med som den refererar till.

Detta innebär att tabellen med en Primary key inte kan förstöras (DROP TABLE) om en annan tabell har en Foreign key som refererar till den. De flesta Databashanterare genererar ett felmeddelande om man försöker.

Auto Increment

En användbar constraint är möjligheten att öka på ett värde automatiskt, exempelvis ett id nummer så att varje ny rad får ett unikt id. Det finns dialektala skillnader på hur detta implementeras.

SQLite = autoincrement. MySQL = auto_increment. SQLserver = Identity(1,1)

```
CREATE TABLE persons (
        id int PRIMARY KEY AUTO_INCREMENT,
        namevarchar(50),
        email varchar(50)
);
```

Använda Databasen

Det vi gjort hittills är den del av sql som kallades **Data DefiningLanguage**. Det används när man skapar och designar databasen. När databasen är up and running så vill vi använda den för att lagra, hämta och manipulera information och nu ska vi titta lite mer på delen av SQL som kallas **Data Manipulation Language.**

Det är framför allt denna del ni kommer ha kontakt med som apputvecklare.

CRUD

CRUD är ett akronym som beskriver de 4 basfunktioner som är nödvändiga för att hantera permanenta lagringsapplikationer (inte minst databaser).

CRUD står för:

Create

Read

Update

Delete

CREATE = INSERT

För att skapa data i en databas så lägger vi in data i tabellerna. Detta görs med kommandot INSERT INTO tabellnamn VALUES (värde_1, värde_2, värde_3…);

(värde_x är värdet som ska in på kolumn x):

Vill man bara fylla i några av kolumnerna specificerar man dessa med dess namn:

INSERT INTO tabellnamn (kolumn_1, kolumn_3) VALUES (värde_1, värde_3);

READ = SELECT

För att läsa information (exempelvis hämta ut till en app) så använder man kommandot SELECT.

SELECT * FROM tabellnamn

Hämtar allt från tabellen ’tabellnamn’

SELECT name, adress FROM tabellnamn

Hämtar innehållet från kolumnerna ’name’ och ’adress’ från tabellen ’tabellnamn’

WHERE

För att filtrera ut specifika rad(er) så använder vi nyckelordet WHERE.

SELECT * FROM tabellnamn WHERE id = 2;

Hämtar innehållet i alla kolumner från raden(raderna) där kolumnen id har värdet 2

SELECT name FROM tabellnamn WHERE salary > 30000;

Hämtar namnet (från kolumnen namn) på alla i tabellen ’tabellnamn’ som har lön som överstiger 30000

WHERE

Vill vi kontrollera fler vilkor så kan vi använda AND och OR

SELECT * FROM tabellnamn WHERE id = 2 AND name = “Bill”;

Hämtar innehållet i alla kolumner från raden(raderna) där kolumnen id har värdet 2 och kolumn name har värdet ”Bill”

SELECT name FROM tabellnamn WHERE id = 1 OR id = 2;

Hämtar namnet (från kolumnen namn) på alla i tabellen ’tabellnamn’ som har id 1 eller 2

LIKE

Vill vi använda WHERE där en kolumn ska innehålla vissa tecken så kan vi använda LIKE och wildcard tecken % och _

SELECT * FROM tabellnamn WHERE name LIKE ‘B%’;

Hämtar innehållet i alla kolumner från raden(raderna) där kolumn name börjar på ’B’

SELECT name FROM tabellnamn WHERE product_code LIKE ’_ _ P _’

Hämtar namnet på alla i tabellen ’tabellnamn’ som är 4 tecken långt och där tredje tecknet är ’P’

UPDATE

Vill vi uppdatera befintlig information så använder vi kommandot UPDATE och nyckelordet SET.

OBS! se alltid till att använda nyckelordet WHERE för att specificera vad som ska uppdateras, annars kommer ALLA rader uppdateras med SET.

UPDATE tabellnamn SET adress = ’gatvägen 5’ WHERE name= ’Bill Palmestedt’;

UPDATE

Vill vi uppdatera flera kolumner så använder vi , .

UPDATE tabellnamn SET namn = ”Bill”, salary = 100, adress = ”vägen 7” WHERE id = 1;

DELETE

För att radera rader använder vi DELETE.

DELETE FROM tabellnamn WHERE id = 1;

Övningar

Som allt annat så blir man inte bättre på något om man inte övar, så det ska vi göra nu

Det finns lite olika online resurser som erbjuder övningar:

https://www.sql-practice.com

https://www.w3schools.com/sql/sql_exercises.asp

https://www.w3resource.com/sql-exercises/