MYSQL - Databaser på riktigt(?) Och början på slutuppgiften

Fram till nu har vi jobbat i en databasFrontend som heter LibreOffice Base. Det är ett bra sätt att få en bild av hur databaser är uppbyggda fungerar. Men verkligheten för utvecklare än inte alltid så snäll att det finns en fungerande frontend. I många fall är jobbet att själv bygga den.
Så, nu ska vi släppa Base en stund och titta på hur en databas fungerar i praktiken - för det är utifrån det vi måste vara beredda att arbeta.
Svårast här brukar vara att få igång MYSQL och får det att rulla. Gör vad ni kan! Får ni inte igång det så kontakta mig så ska jag se vad jag kan göra för att hjälpa!

När det rullar

Så är allt man ser en terminal - och det är här vi kommer att jobba. Alla kommandon vi skriver avslutas med ; men det går utmärt att dela upp kommandona på flera rader - vilket gör det väldigt mycket lättare att skriva långa queries.
Nej, man behöver inte kunna alla SQL-kommandon utantill (Om man inte vet att det är databaser man vill jobba med) däremot ska man kunna hitta och använda dem. Det är FÅ arbeten inom data där ni aldrig kommer i kontakt med dem. Vänj er vid att skriva dem! Det kommer bli tufft framöver om man måste googla för att skriva en enkel sökning.

Att göra en databas

- SHOW DATABASES; Visar en lista över de databaser som vi har tillgängliga.
- CREATE DATABASE mybase; skapar en databas med namnet mybase
- DROP mybase; tar bort databasen med namnet mybase
- USE mybase växlar till mybase

För att komma igång: Skapa en ny databas med ett valfritt namn och växla till den.
En databas består ju av tabeller. Nu bygger vi en sådan med det fantasifulla namnet 'first_table'.

- Nu skriver vi:
  CREATE TABLE first_table(Id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(255), phone VARCHAR(255));
  Gör i mångt och mycket allt det vi gjorde i Skapa tabellvyn". Notera hur vi anger primary key. Observera att det är fullt möjligt att INTE ha med PK. Men som vi vet nu så måste den vara där för att vi ska kunna länka våra tabeller!

- Kör vi nu kommandot: SHOW TABLES; så kan vi se vår skapade tabell.

- För att få mer information om en tabellen kan vi skriva:
  DESCRIBE first_table;

Lägga till data i en tabell

- För att lägga in en post skriver vi:
  INSERT INTO first_table (name, phone) VALUE (”RUDOLF RUSK" ,"08-111111");
  Eftersom vi ovan satte AUTO_INCREMENT på vår primary key så behöver vi inte lägga något värde i Id-kolumnen.

- Nu kan vi kolla att det fungerade genom att skriva:
  SELECT * FROM first_table;

- Men vad nu! Vi har ju skrivit fel telefonnummer på Rudolf! Han har ju 08-222222. Då ändrar vi det!
  UPDATE first_table set phone = ”08-222222” where Id = 2;
  Vi kollar att det fungerade med :
  SELECT * FROM first_table;

- Rackarns, nu kom vi på att Rudolf Rusk ju inte alls ska vara med i vår tabell! Ingen fara, vi skriver bara:
  DELETE FROM first_table where Id =1;
  Och vi kan kolla med: SELECT * FROM first_table;

- När vi tänker efter vill ha Rudolf där - lägg tillbaka honom i tabellen.

Modifiera en tabell

Tidigare i kursen hade vi ett exempel där en anställd kunde ha flera telefoner. Vi löste det genom att ha ett telefonID som länkade till en telefonlista. För att fixa till det vill vi ha integers istället för varchar i telefonkolumnen:

- Kommandot för att ändra i en tabell är ALTER TABLE vi börjar med att ta bort telefonKolumnen:
  ALTER TABLE first_table DROP phone;

- för att lägga till en ny kolumn så kör vi:
  ALTER TABLE first$_table ADD phone int$; *Eller* :
  $ALTER TABLE first_table ADD phone int AFTER name$;

En ny kolumn lägg automatiskt sist. Lägger vi ett AFTER så kan vi bestämma var vi vill placera den (I det här fallet är det ju vilket som - men testa gärna att sätta in ännu en spalt mellan name och phone och ta bort den igen som övning.

- Nu ger vi Rudolf ett ett värde i phone som vi ska länka till en telefontabell:
  UPDATE first_table SET phone = 1 WHERE name = ”Rudolf Rusk”;

En tabell till och lite länkande

För att snabba ut sökande och länkande så använder man index (Läs mer här: https://blog.k.io/viaduct/mysql-indexes-primer) För oss betyder det att vi för att kunna länka en spalt behöver skapa ett index för spalten vi länkar från.

- Vi skriver:
  CREATE INDEX phoneIndex ON first_table (phone);

- Sen kan vi skapa den tabell vi vill länka till tbl_phone:
  CREATE TABLE tbl_phone
  (PhoneID int, PhoneNumber varchar(255),
  PRIMARY KEY(PhoneID,PhoneNumber), FOREIGN KEY (PhoneID)
  REFERENCES first_table(phone)
  ON DELETE CASCADE);
  Puh! Vi skapar tabellen och vi skapar en Composite primary key och vi länkar via PhoneID till first_table. Och vi lägger Delete cascade -vilket är precis samma sak som vi gjorde i Base. Nu är våra tabeller länkade och tar vi bort Ruskige Rudolf så försvinner alla hans telefonnummer.

- Sätter vi in Rudolf telefonnummer:
  INSERT INTO tbl_phone (PhoneID, PhoneNumber) VALUES (1,”08-111111”);

INSERT INTO tbl_phone (PhoneID, PhoneNumber) VALUES (1,"08-222222");

Sökning och listning

Nu har två länkade tabeller! Nu kan vi börja söka och leka med SQL:

- Vi sätter en person till utan telefonnummer:
  INSERT INTO first_table (name, phone) VALUE ("Benny Burk", 2):
- nu har (i alla fall) lite mer att jobba med låt oss slå ihop våra tabeller testa: SELECT * FROM first_table, tbl_phone;
  Vad är det vi ser?
- Hur får vi ut vad vi vill ha? Inner Join hjälper oss att länka: SELECT * FROM first_table INNER JOIN tbl_phone on first_table.phone = tbl_phone.PhoneID;
  INNER JOIN betyder att vi få ut de poster där first_table.phone matchar tbl_phone,PhoneID.
- Lek nu vidare med dina tabeller. Testa att lägga in fler världen och se att du kan göra de sökningar du vill och behöver!
  W3schools ger en bra översikt över SQL- kommandon:
  https://www.w3schools.com/sql/

Slutuppgift del 1

Använd nu dina kunskaper i MySQL och lägg in och länka dina (så klart normaliserade) tabeller från Inlupp 1 inklusive samband. Lägg tid och gör det ordentligt. Det är den här databasen som vi kommer använda java för att arbeta mot.