MySQL Laboration 3

Denna lab går ut på att vi ska jobba mycket med att skapa och definiera kolumner i tabeller, samt skapa relationer mellan dessa. Det viktigaste att hålla koll på då är något som heter **primärnyckel**, eller **primary key** på engelska.

De flesta tabeller som skapas i databaser innehåller en kolumn som är definierad som just en **primärnyckel**. Med **primärnyckel** menas att alla värden i den kolumnen måste vara **unika**. *Inget värde kan vara det andra likt!* Detta är för att man alltid ska ha ett värde som skiljer en rad från alla andra.

Nedan kommer ett litet exempel för att förtydliga hur det kan vara användbart!

Tänk er att vi har en tabell med personer som ser ut på detta vis: *(Samma struktur som lab1)*

Personer		
Förnamn	**Efternamn**	**Ålder**
Henrik	Efson	29
Tomas	Weis	48
Stefan	Zethraeus	25
Joakim	Ohlsson	35

Ifrån denna tabell går det jättebra att hämta eller ändra information om någon som heter "Henrik" eller "Stefan". Men tänk om vi ska lagra data om hundratusentals kunder, då kommer många garanterat ha samma förnamn och högst troligt samma efternamn och vissa även samma ålder.

Frågor skickas till Victor Bohmies på Progress eller via mail: victor.bohmies@innovationsgymnasiet.se

Det kan exempelvis se ut såhär om man hämtar alla personer som heter "Joakim Ohlsson":

Personer		
Förnamn	**Efternamn**	**Ålder**
Joakim	Ohlsson	29
Joakim	Ohlsson	48
Joakim	Ohlsson	25
Joakim	Ohlsson	25

Nu blir det ju jobbigt om vi vill ändra eller hämta information om en specifik kund som heter "Joakim Ohlsson". Speciellt när vi har två stycken som är lika gamla. Då hade det varit bra att ha just en primärnyckel som gör att varje rad blir unik.

Det kan då se ut såhär:

Personer			
Id	**Förnamn**	**Efternamn**	**Ålder**
1	Joakim	Ohlsson	29
2	Joakim	Ohlsson	48
3	Joakim	Ohlsson	25
4	Joakim	Ohlsson	25

Nu går det alldeles utmärkt att ändra eller hämta information om den "Joakim Ohlsson" som har exempelvis Id=4. Eller vilken person som helst för den delen, för nu har alla ett unikt värde som gör varje rad*(person)* urskiljbar från de andra.

Här har jag även länkat en video som förklarar lite om primary keys ifall ni föredrar att ha det förklarat av någon som pratar (den är på engelska).

Inlämning sker på Progress.

Frågor skickas till Victor Bohmies på Progress eller via mail: victor.bohmies@innovationsgymnasiet.se

LABORERA SJÄLV:

Nu ska vi skapa en tabell som heter ***kunder*** och den får innehålla en kolumn som heter **kundnummer** som ska vara en **primärnyckel**.

> ***Det skrivs så här:***
>
> **CREATE TABLE kunder(Kundnummer int(8) PRIMARY KEY AUTO_INCREMENT);**

Koden utläses så att vi skapar en tabell som heter ***kunder*** och den får en kolumn som heter **kundnummer** och är av datatypen **int***(heltal)*. Kolumnen är en **PRIMARY KEY** vilket innebär att databasen inte kommer acceptera att man lägger in dubbla värden i den kolumnen och vi lägger även till **AUTO_INCREMENT** som gör att vi inte behöver bry oss om att kolla vilket unikt **kundnummer** varje ny rad ska få, utan tabellen skapar ett nytt **kundnummer** som är ett högre än det som redan finns i tabellen!

Nu kan vi lägga till flera kolumner till tabellen.

> ***En för förnamn och en för efternamn:***
>
> **ALTER TABLE kunder ADD (Förnamn varchar(32));**
>
> **ALTER TABLE kunder ADD (Efternamn varchar(32));**

Sen kanske vi vill ha personnummer på våra kunder också. Och alla vet att personnummer i världen är unika så att ha personnummer som en primärnyckel kan ju kanske vara ett smidigt alternativ?

Tyvärr så är det inte så bra. Detta eftersom personnummer är känslig information och borde i många fall vara skyddad i form av kryptering så det är ingen bra ide att ha personnummer som primärnyckel. Men det finns en annan nyckel som heter **UNIQUE** och det menas helt enkelt att en kolumn som har nyckeltypen **UNIQUE** bara får innehålla unika värden.

> ***Vill vi skapa en kolumn för personnummer som ska ha en unik nyckel kan vi skriva så här:***
>
> **ALTER TABLE kunder ADD (Personnummer bigint(10) UNIQUE);**

Observera att vi måste skriva bigint(10) då en vanlig int i databaser max kan innehålla tal upp till och med ett värde av 99999999 *(8st siffror, därav int(8))*.

Inlämning sker på Progress.

Alltså: lägger vi till ordet **UNIQUE** efter datatypen på en kolumn så kommer den få nyckeltypen **UNIQUE**. Samma som när vi lade till **PRIMARY KEY** efter datatypen, som också är en typ av nyckel.

I detta fall skapar vi en kolumn som kommer innehålla siffror i ett personnummer men det vanligaste är att man gör det till en lång sträng som motsvarar ett krypterat värde för ett personnummer!

Man kan även skapa alla dessa kolumner i ett kommando, istället för att skriva flera rader som vi gjort ovan.

Det ser då ut så här:

```
CREATE TABLE kunder(Kundnummer int(8) PRIMARY KEY AUTO_INCREMENT, Förnamn varchar(32), Efternamn varchar(32), Personnummer bigint(10) UNIQUE);
```

Nu vill vi skapa en ny tabell som får heta ***fakturor*** och den ska på något sätt vara ihopkopplad med ***kunder*** så att man kan med hjälp av en kunds ***kundnummer*** kan hämta information om kundens ***fakturor***. Vi börjar med att skapa kolumner för det vi behöver veta om varje faktura. I detta fall vill jag lagra data för fakturanummer, totalbelopp och kontonummer att betala beloppet till.

```
CREATE TABLE fakturor (Fakturanummer int(8) PRIMARY KEY AUTO_INCREMENT);
ALTER TABLE fakturor ADD (Totalbelopp float);
ALTER TABLE fakturor ADD (Kontonummer int(8));
```

Nu har vi använt en främmande datatyp som vi inte stött på tidigare, nämligen **float**.

Datatypen **float** är lika som i programmering, ett tal som kan innehålla decimaler, inga konstigheter egentligen.

Men hur ska vi nu koppla ihop tabellerna ***fakturor*** och ***kunder***?

Jo, vi får lägga till en kolumn som heter ***kundnummer*** och den ska vara kopplad till tabellen ***kunder*** och dess kolumn ***kundnummer***.

Det skriver vi så här:

```
ALTER TABLE fakturor ADD (Kundnummer int(8));
```

Inlämning sker på Progress.

Frågor skickas till Victor Bohmies på Progress eller via mail: victor.bohmies@innovationsgymnasiet.se

```
ALTER TABLE fakturor ADD FOREIGN KEY
(Kundnummer) REFERENCES kunder(Kundnummer);
```

Nu är tabellerna ihopkopplade med en **FOREIGN KEY**, det innebär att kolumnen ***kundnummer*** i tabellen ***fakturor*** är en främmande nyckel från en annan tabell och i detta fallet är det ju från kolumnen ***kundnummer*** i tabellen ***kunder***.

Det innebär lite nya saker för dessa två tabeller. Exempelvis så kommer tabellen ***fakturor*** att säga ifrån om vi försöker skapa en faktura till en kund med ett ***kundnummer*** som inte existerar.Den kommer även klaga ifall vi försöker ta bort en kund från ***kunder*** och den kunden har en eller flera fakturor i tabellen ***fakturor***. För det får inte existera en faktura till en kund som inte finns i tabellen ***kunder***.

En **FOREIGN KEY** mellan två kolumner innebär med andra ord att vi talar om för databasen att dessa två kolumner är besläktade med varandra och kan enbart existera tillsammans. Allt annat vi försöker göra som bryter mot den regeln kommer databasen att avvisa!

En vanlig sak är att göra är att när man skapar en rad är att lagra exakt den tidpunkt då raden lades till i tabellen. Det kan vi göra genom att skapa en kolumn som får i uppdrag att lagra just den informationen. En sådan kolumn ska vara av datatypen **timestamp** och ha ett **DEFAULT** värde som sätts till **CURRENT_TIMESTAMP**.

Det kan skrivas så här:

```
ALTER TABLE fakturor ADD (Skapad timestamp
DEFAULT CURRENT_TIMESTAMP);
```

Datatypen **timestamp** används för att lagra en viss tidpunkt och alla databaser har en inbyggd klocka som håller koll på vilken tidpunkt, dag och år det är just för stunden.

Ett **timestamps** format är uppbyggt på detta vid: **"YYYY-MM-DD HH:MM:SS"**. **timestamps** går utmärkt att jämföra med varandra som om vi skulle jämföra vanliga datum/tider på en tidsskala. Använda bara större än och mindre än! (> . <)

DEFAULT kan man lägga till på alla kolumner man skapar och det man skriver efteråt kommer att bli ett förinställt värde som varje ny rad som man skapar kommer att tilldelas om inget annat specificeras! Om man inte har ett **DEFAULT**-värde kommer det alltid vara **NULL**.

NYA KOMMANDON OCH FUNKTIONER:

NOW() - Har ett värde som motsvarar den aktuella tidpunkten vid utförandet av frågan.

Exempel: Hämta alla fakturor som är skapade före denna tidpunkt.

Inlämning sker på Progress.

Frågor skickas till Victor Bohmies på Progress eller via mail: victor.bohmies@innovationsgymnasiet.se

SELECT * FROM Fakturor WHERE skapad < NOW() ;

TIMESTAMPADD(a,b,c) - Används för att kunna räkna med **timestamps**.

c motsvarar en tid av typen **timestamp** som du vill modifiera
a motsvarar vad du vill ändra **c** med.***YEAR,MONTH,DAY,HOUR,MINUTE,SECOND***
b motsvarar hur många av **a** du vill ändra **c** med. Kan vara positivt och negativt.

Exempel: För att skapa ett timestamp med en tidpunkt som är 11 dagar innan den 11 september 2011 kan man skriva såhär:
TIMESTAMPADD(DAY,-11,"2001-09-11");
Observera att i detta fall kommer tiden sättas till 00:00:00 eftersom vi ej specificerat tidpunkt.

OFFSET x - Kommer att avsätta/ignorera de första x antal värden i ditt resultat.
Exempel: Vill du ha den näst största av något kan du skriva:
SELECT * FROM *tabell* ORDER BY *kolumn* LIMIT 1 OFFSET 1;

Observera att **OFFSET** enbart fungerar i samband med **LIMIT**!

Inlämning sker på Progress.

Frågor skickas till Victor Bohmies på Progress eller via mail: victor.bohmies@innovationsgymnasiet.se

INLÄMNINGSUPPGIFT:

Ni ska skapa en databas för en blogg där tanken är att man ska kunna registrera en användare, skapa blogginlägg och kommentera på andras blogginlägg. Man ska även kunna gilla kommentarer och inlägg.

Ni ska även fylla på databasen med ett antal rader och svara på frågorna under del 3.

1. SKAPA TRE TABELLER I DIN DATABAS MED TILLHÖRANDE KOLUMNER.

USERS (TABELL FÖR ATT LAGRA ANVÄNDARE)
ID
Username
First_name
Last_name
Mail

POSTS (TABELL FÖR ATT LAGRA INLÄGG SKAPADE AV **EN** SPECIFIK ANVÄNDARE)
ID
User_ID
Headline
Text
Likes
Creation_time

COMMENTS (TABELL FÖR ATT LAGRA KOMMENTARER PÅ **ETT** SPECIFIKT INLÄGG SKAPADE AV **EN** SPECIFIK ANVÄNDARE)
ID
Post_ID
User_ID
Text
Likes
Creation_time

OBS! Jag vill veta den kod du skrev för att skapa dessa tabeller! (CREATE …), Så var noggrann med att dokumentera.

2. FYLL PÅ DINA TABELLER MED INFORMATION.

A. Skapa tre fullständiga användare.
B. Skapa två fullständiga inlägg per användare.
C. Skapa minst två kommentarer per användare på olika posts.

Inlämning sker på Progress.

Frågor skickas till Victor Bohmies på Progress eller via mail: victor.bohmies@innovationsgymnasiet.se

OBS! Jag vill veta den kod du skrev för att skapa dessa rader! (INSERT…), Så var noggrann med att dokumentera.

3. HÄMTA/MODIFIERA INFORMATION I DATABASEN:

A. Vad ska jag skriva för fråga om jag vill få den användare med flest likes på alla sina inlägg?
B. Vad ska jag skriva för fråga om jag vill få fram det tredje första inlägget som någonsin skrevs?
C. Vad ska jag skriva för fråga om jag vill hämta antal kommentarer varje användare skrivit?
D. Vad ska jag skriva för fråga om jag vill hämta id på den post med näst-flest kommentarer?
E. Vad ska jag skriva för fråga för att få veta hur många inlägg och kommentarer som skrivits?
F. Vad ska jag skriva för fråga för att hämta alla inlägg som är skapade i September 2015?
G. Vad ska jag skriva om någon har gillat kommentaren med id=4? *(Jag vill då öka Likes med 1)*
H. Vad ska jag skriva för att ändra på det senaste inlägget användaren med id=1 skrev?
I. Vad ska jag skriva för att ta bort alla kommentarer som användaren med id=3 skrivit?
J. Vad ska jag skriva för att ta bort användaren med id=2? *(Flera rader krävs)*

Inlämning sker på Progress.