Laboration 6, Databashantering med MySQL - Modellering

Av: Marcus Rejås <marcus@rejas.se>

> *I denna labb skall vi för omväxlings skull inte jobba med datorn. Vi skall istället titta på hur datamodellering går till. Vi skall lära oss en enkel form av modellering och skall även praktiskt utföra den.*

Från förra laborationen

Vi kommer inte att använda någonting från föregående laborationer. Man kan om man vill göra denna laboration som första laboration men jag rekommenderar att man gör dem i ordning. Man tillgodogör sig denna laboration lättare om man har gjort de andra.

Datamodellering

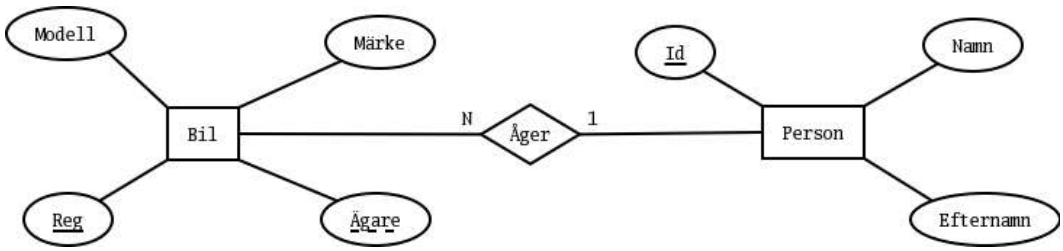
I nästan alla vetenskaper, inom industri och snart sagt överallt använder man sig av modeller. Det gör man även inom databashanteringen. Modeller använder man som en avbild av en verklighet för att se hur den beter sig i olika situationer. Man använder också modeller för att se hur någon produkt eller annat alster kommer att se ut och fungera när det är klart. Ofta bygger man modeller för att:

- De är billigare än den verkliga produkten eller miljön.
- De är mindre eller större än den verkliga produkten eller miljön. Det är lättare att ändra och testa saker på en modell som är mindre än tunneln under engelska kanalen eller större än en atomkärna.
- De är inte lika farliga som den riktiga miljön.
- De är lättare att visa upp eller få en helhetsbild över.
- De är enklare att ändra än den riktiga miljön.

Inom databashanteringen är det i huvudsak de två sista argumenten som gör att jobbar med modeller. Om man skall göra en databas är det enklare att starta med en modell på en whiteboard eller papper innan man börjar skapa tabeller och liknande i MySQL eller vilken databashanterare man nu valt. Det finns standarder för hur modellering skall gå till men vi kommer att använda en lite förenklad variant. Den variant vi använder i denna laboration liknar en modell som heter ER. Den stämmer dock inte helt överens. Är du intresserad av att lära dig ER-modellering ”på riktigt” så rekommenderar jag att du läser följande länk där det står väldigt bra förklarat hur ER-modellering fungerar. Den kan dock ses som överkurs.

```
http://www.ida.liu.se/~tompa/databaser/er/
```

Nedan följer ett exempel på en mycket enkel datamodell. Titta tillbaka på den och lägg den på minnet. Titta sedan på den igen när du läser resten av denna text och när du gör uppgifterna.



Figur 1: Exempel på en enkel datamodell

Identifiera objekten (entiteterna)

När man skapar en modell så utgår man oftast från den verklighet vi känner. Skall vi skapa en datamodell över till exempel ett företag så börjar vi med att ta reda på vilka objekt i detta företag som skall vara med i vår datamodell och i förlängningen i vår databas. Objekten är sådant som går att ta på (man kanske inte kan ta på dem alltid, men det är oftast substantiv). Exempel på objekt kan vara faktura, bil, person, räkning, m.m. Objekten kallas ibland för entiteter. Dessa blir sedan oftast tabeller i den färdiga databasen.

Normalt sätt så ritar man objekten som fyrkanter med namnet i mitten. Om vi skall göra en databas över bilar och vilka som äger bilarna kan bra objekt vara **bil** och **person**.

Attribut

Varje objekt (eller entitet som det ju också kallas) har ett visst antal attribut. Attribut är olika egenskaper som det aktuella objektet har. Det kan vara till exempel att bilen är röd, den har ett visst registreringsnummer eller den är av ett visst märke eller en viss modell. Alla dessa saker är attribut. Vi ritar attributen som ovaler med namnet i mitten och ett streck till det objekt attributet hör. Det kan finnas flera attribut med samma namn i modellen men de måste vara anslutna till olika objekt. Ett objekt kan alltså inte ha två attribut med samma namn.

Nycklar

Ett attribut som på något sätt unikt identifierar en instans av ett objekt kallas för en nyckel. Det vill säga om jag med hjälp av ett enda attribut säkert kan peka ut en enda post av objektet så är det en nyckel. Till exempel så kan ett registreringsnummer på en bil användas för att hitta just en bil. Det är ingen risk att det finns två bilar med samma registreringsnummer. Registreringsnummer är i det fallet en bra nyckel.

Alla objekt bör ha en nyckel. Har objektet ingen naturlig nyckel att använda så kan enkelt skapa en egen nyckel med till exempel ett löpnummer. Ofta kallas dessa för ”id”.

Ibland kan det vara bra att man, trots att det finns en naturlig nyckel, ändå skapar en egen. Detta används ofta inom till exempel arbetsplatser och inom föreningar. Alla personer i Sverige har ju ett personnummer, ändå får man nästan alltid ett medlemsnummer eller anställningsnummer när man läggs upp i en medlemsdatabas eller en databas på ett företag. Det gör man för att alla kanske inte vill att deras personnummer skall finnas med i databasen. Genom att skapa en annan nyckel kan man stoppa in även personer som inte vill dela med sig av sitt personnummer i databasen. Man kan ändå ha personnummer som ett vanligt attribut om man vill det.

Det attribut som är nyckel stryker man under med en heldragen linje.

Relationer

Relationer inför man för att man vill visa hur de olika objekten är relaterade till varandra. I exemplet ovan är till exempel är objektet bilar relaterat till personer. Ofta vill man sätta namn på relationer med, då skulle man kunna kalla denna relationen för ”äger”, det vill säga en person äger en bil. Vi markerar ut om det är fråga om en 1:1-, 1:N- eller N:N-relation. I en N:N relation måste ett kopplingsobjekt skapas.

Ett exempel på ett kopplingsobjekt kan vara följande.

Antag att du skall göra en databas åt ett flygbolag. Ett flygbolag har ett antal flygkaptener och ett antal flygplan. Flygkaptenerna kan flyga flera plan och ett plan kan flygas av flera kapterner. Alltså har vi en N:N-relation. Men vi vet att en kapten kan bara flyga ett flygplan i taget. Varje sådant tillfälle brukar kallas en flight. En flight binder samman en kapten med ett flygplan och dessutom kan man lägga till extra information i objektet flight, till exempel tider, destinationer och annat. Tittar man på relationerna nu så ser vi att en kapten kan vara på många flighter men en flight kan bara ha en kapten (1:N) precis samma sak gäller flygplanen. Nu har vi ingen N:N-relation utan bara 1:N-relationer. Eftersom vi vet att den främmande nyckeln skall vara på många-sidan har vi inga problem att sätta ut den.

Främmande nycklar

Normalt sett så skriver man inte ut främmande nycklar när man modellerar. Dessa kan man ju lista ut i efterhand vilka de skall vara. Men när man vet att man modellerar för en relationsdatabas som vi gör så är det lika bra att rita ut dem. Tänk på att den främmande nyckeln alltid skall vara på många-sidan (N) av en relation. Ett sätt att märka ut en främmande nyckel är att stryka under den med en streckad linje.

Att svara på

I denna labb så svarar du på separata blad som du lämnar in till laborationshandledaren. Ni jobbar en och en. Glöm inte att skriva namn på det du lämnar in.

Uppgift 1

I denna uppgift skall du göra en databas för ett bokförlag. Förlaget vill kunna lägga in de böcker de har men också olika författare. Du börjar förstås med att göra en datamodell. Du gör också den lilla förenklingen att en bok bara kan ha en författare. Din datamodell, den är uppgift 1.

Uppgift 2

Lägg till i din modell från den förra uppgiften att en bok kan ha flera författare.

Uppgift 3[*]

Den här gången skall du göra en databas till ett bussbolag. Bussbolaget vill ha en databas med alla deras bussar och alla deras chaufförer. De vill dessutom hålla reda på vilken chaufför som kör vilken buss vid ett bestämt tillfälle. Börja som tidigare med att identifiera objekten. Forstätt sedan med relationerna och se sedan om du fått med allt. Din datamodell lämnar du in. Skriv gärna en liten text också med förklaringar där det kan vara nödvändigt.

* Uppgift som är lite svårare

