Normalisering – exempel

Normalisering är som nämnts tidigare en teori för relationsdatabaser som kan användas för att kontrollera att befintliga tabeller i en databas har en bra design. Genom att normalisera en tabell kan vi undvika problem som redundant data och problem som kan uppstå vid insättning och borttagning av data. Det finns ett antal olika normalformer som en tabell kan uppnå och de olika normalformerna innebär kort vilken grad av kontroll som utförs på tabellen. För att avgöra vilken normalform en tabell befinner sig i är det viktigt att vi har identifierat alla fullständigt funktionella beroenden (ffb) i tabell. Därför är det väldigt viktigt att ha bra koll på vad funktionella beroenden (fb) innebär.

Funktionella beroenden

Definition av fb och ffb lyder som följer:

Om A och B är attribut i en relation R så är B fb av A (skrivs A --> B) om det för varje värde på A finns ett och endast ett tillhörande värde på B. A och B kan var för sig bestå utav flera attribut. Om B är fb av A så följer att B är ffb av A om B även är ej fb av någon delmängd till A.

Detta brukar för många låta som rena rappakalja och för att förenkla och förtydliga detta lite kan vi utgå från den beskrivning som görs i webbkursen om databaser. Vi har ett funktionellt beroende om värdet på ett attribut, eller en kombination av attribut, A bestämmer värdet på ett annat attribut B. Attributet B är då funktionellt beroende av A. Det vill säga om A på flera rader i tabellen har samma värde, måste också värdena på B vara samma. Då har vi ett funktionellt beroende A --> B. A kallas för determinant och determinerar B (bestämmer värdet på B).

Ett fullständigt funktionellt beroende (ffb) är ett funktionellt beroende där man inte kan ta bort några attribut ur determinanten om det fortfarande ska vara ett funktionellt beroende. Ett annat sätt att säga det är att determinanten ska vara minimal. Detta kan vi delvis jämföra med diskussionen om supernycklar och kandidatnycklar (där en kandidatnyckel är en minimal supernyckel, det vill säga en supernyckel där man inte kan ta bort några attribut om den fortfarande ska vara garanterat unik.)

Har vi ett fb där determinanten består av enbart ett attribut är det alltid frågan om ett ffb eftersom vi inte kan ta bort något attribut från determinanten. Det är endast när determinanten består av flera attribut det kan finnas risk att det är ett fb men inte ett ffb.

För att reda ut skillnaderna ytterligare utgår vi från tabellen Medlem nedan som ett exempel och försöker hitta alla fullständigt funktionella beroenden. Primärnyckel i tabellen är personnr.

Medlem

personnr	**namn**	**adress**	**postnr**	**postort**
111111	Kalle	Gatan 3	322 21	Kruv
222222	Stina	Vägen 23	122 45	Tråka

333333	Kalle	Gatan 4	322 21	Kruv
444444	Ida	Storvägen 2	432 12	Grytan
555555	Sara	Storvägen 2	555 12	Erbode

För att hitta vilka funktionella beroenden som finns i en tabell tycker jag det är enklast att leta efter vilka attribut som är determinanter. Vi kan direkt säga att personnr är en determinant, detta eftersom personnr är primärnyckel i tabellen och då alltid determinerar alla andra attribut. På varje rad i tabellen där det i attributet personnr står värdet 111111, kommer det alltid att på motsvarande rader i tabellen att stå värdena Kalle, Gata 3, 322 21 och Kruv i attributen namn, adress, postnr och postort.

Här har vi alltså ett funktionellt beroende mellan attributen personnr samt namn, adress, postnr och postort. De senare attributen är funktionellt beroende av personnr. Detta funktionella beroende skriver vi så här:

personnr --> namn, adress, postnr, postort

Är detta funktionella beroende även ett fullständigt funktionellt beroende? Ja, eftersom vi från vänstra sidan av pilen (determinanten) inte kan ta bort några attribut. Determinanten personnr är minimal.

Ett annat funktionellt beroende i tabellen är följande:

personnr, namn --> adress, postnr, postort

På varje rad i tabellen där det i attributen personnr och namn står värdena 111111 och Kalle, kommer det alltid att på motsvarande rader i tabellen att stå värdena Gata 3, 322 21 och Kruv i attributen adress, postnr och postort.

Detta är dock inget fullständigt funktionellt beroende eftersom vi från determinanten kan ta bort attributet namn och beroendet fortfarande gäller. Det vi är intresserade av när vi normaliserar är alltid de fullständigt funktionella beroendena.

För att leta vidare efter fler fullständigt funktionella beroenden kan vi undersöka om attributet namn en determinant? På varje rad i tabellen där det för attributet namn står till exempel värdet kalle, kommer det att på motsvarande rader i andra attribut alltid att stå samma värde? Nej, eftersom det till exempel i attributet adress förekommer olika värden för Kalle. Som vi ser så står det i attributet namn värdet Kalle på två rader i tabellen. På motsvarande rader i tabellen står det i attributet adress inte samma värde. På en rad står det Gatan 3 och på en annan rad står det Gatan 4. Attributet namn determinerar därmed inte attributet adress eftersom det för samma värde i namn (Kalle) står olika värden i adress (Gatan 3 och Gatan 4). Enligt samma resonemang finner vi även att namn inte determinerar några andra attribut i tabellen.

Samma sak gäller attributet adress där vi med säkerhet inte kan säga att för varje värde i adress kommer det att stå samma värden i övriga attribut. T.ex. kan det finnas flera Storvägen 2 i Sverige fast med olika postnummer och postorter.

Om vi däremot tittar på attributet postnr finner vi ett samband mellan värdena i postnr och postort. För varje värde i postnr står det samma sak i postort. Till exempel förekommer värdet 322 21 på två rader i attributet postnr och på motsvarande rader i attributet postort står det alltid Kruv. Med andra ord är postnr en determinant för den determinerar postort (bestämmer värdet på postort).

Däremot determinerar postort inga andra attribut. På varje rad där det i postnr står värdet 322 21 kan vi inte säga att det står samma värde i till exempel namn (visserligen gör det så nu men i det stora hela borde det det rimligtvis kunna finnas flera personer med namnet Kalle på en postort?). Vi har därför följande fullständigt funktionella beroende:

postnr --> postort

I mina exempel ovan har jag enbart undersökt determinanter som består av ett attribut. När vi normaliserar är det även viktigt att leta efter determinanter som består av två eller flera attribut. Det är helt ok med determinanter som innehåller flera attribut så länge som det är frågan om en minimal determinant.

De fullständigt funktionella beroenden som existerar i tabellen Medlem är följande:

ffb1: personnr --> namn, adress, postnr, postort
ffb2: postnr --> postort

Det är vanligt att vi numrerar våra funktionella beroenden så att vi enklare kan referera till dem i en löpande text.

Första normalformen (1NF)

Den första normalformen säger att det endast får finnas ett värde i varje attribut. Det får inte finnas en lista med värden. Låt oss ta namn som exempel. Ett namn kan antingen vara Robert eller Robert Jonsson. Lagrar vi detta i en databas så är det fortfarande bara ett värde oavsett om namnet bara består av förnamnet eller om det består av både för- och efternamn.

Det samma gäller för adress. Vi kan till exempel ha datatypen text för att lagra en adress i en databas. Då spelar det ingen roll om adressen består av både gatuadress, postnummer och postort. Det är fortfarande bara en adress som lagras. Vi kan lagra strängen "Knutvägen 11, 345 54 Knutfors" i attributet adress utan problem. Om vi däremot försöker lagra två adresser i samma attribut bryter vi mot 1NF. Vår tabell Medlem bryter inte mot 1NF eftersom det bara finns ett värde i varje cell.

Andra normalformen (2NF)

2NF säger att det inte får finnas några partiella determineringar. En partiell determinering uppstår när vi har en sammansatt primärnyckel (eller sammansatt kandidatnyckel) och där

ett av attributen ensamt determinerat ett annat attribut. Ett annat sätt att uttrycka det på är att det inte får finnas några pilar som går från en del av en kandidatnyckel utan endast hela.

Eftersom primärnyckeln i tabellen Medlem endast består av ett attribut kan det inte finnas några determineringar som går från en del av primärnyckeln. Finns det determineringar som går från en del av någon kandidatnyckel? Vi kan alltså inte bara ta hänsyn till primärnyckeln när vi normaliserar till 2NF utan måste även beakta alla kandidatnycklar i tabellen. Vilka kandidatnycklar finns det då?

Namn är ingen kandidatnyckel eftersom det kan finnas flera olika personer med samma namn. Adress är ingen kandidatnyckel eftersom det kan finnas flera olika personer som bor på samma adress. Postnr och postort är av samma anledning inte heller några kandidatnycklar.

Finns det kombinationer av attribut som bildar en kandidatnyckel? Ja, utifrån de exempeldata som är visas i tabellen verkar det som att namn tillsammans med adress utgör en kandidatnyckel. Vissa tveksamheter finns dock. I verkligheten är det väl så att det kan finnas flera personer som heter Kalle som bor på en och samma adress? Därför är kombinationen namn och adress egentligen ingen kandidatnyckel. Men för exemplets skull antar vi att namn och adress utgör en kandidatnyckel.

Vi har då följande fullständigt funktionella beroenden:

ffb1: personnr --> namn, adress, postnr, postort
ffb2: postnr --> postort
ffb3: {namn, adress} --> personnr, postnr, postort

Detta bryter dock fortfarande inte mot 2NF eftersom varken namn eller adress ensamt determinerar något annat attribut. Det vill säga det går inga pilar från en del av kombinationen namn, adress utan enbart från hela.

Om vi däremot gör följande långsökta antagande att det inom olika postnummerområden inte kan finnas olika personer med samma namn, Det vill säga om namnet Kalle förekommer inom postnummerområdet 322 21 kan det i andra postnummerområden, till exempel 122 45, inte finnas personer med namnet Kalle. Ett visst namn får endast förekomma inom ett visst postnummer. Givetvis stämmer detta inte med verkligheten men vi har nu även följande fullständigt funktionella beroende:

ffb1: personnr --> namn, adress, postnr, postort
ffb2: postnr --> postort
ffb3: {namn, adress} --> personnr, postnr, postort
ffb4: namn --> postnr

Nu har vi en tabell som inte uppfyller kraven på 2NF. Vi har attributet namn, som är en del av kandidatnyckeln namn, adress, men som ensamt determinerar ett annat attribut. Här skulle vi behöva dela upp tabellen Medlem till två nya tabeller (enligt tillvägagångssättet som beskrivs under 3NF).

Fortsättningsvis utgår vi återigen endast från att ffb1 och ffb2 är de fullständigt funktionella beroenden som existerar i tabellen.

Tredje normalformen (3NF)

3NF säger att det inte får finnas några transitiva determineringar. En transitiv determinering har vi när ett attribut, som inte ingår i någon kandidatnyckel, determinerar ett annat attribut, som inte heller ingår i någon kandidatnyckel. Eller på ett annat sätt, det får inte finnas några pilar som går mellan attribut utanför kandidatnycklarna, utan endast en pil från en kandidatnyckel eller en pil till en kandidatnyckel.

Återigen är det alltså viktigt att ha koll på vilka kandidatnycklar som finns i tabellen vi normaliserar för att avgöra vilken normalform tabellen befinner sig i. En kandidatnyckel är som du känner till sen tidigare ett attribut eller en kombination av attribut vars värden är unika (av alla kandidatnycklarna väljer vi sen en som primärnyckel). I tabellen Medlem är personnr primärnyckel och då även en kandidatnyckel. Är namn en kandidatnyckel, det vill säga är värdet i attributet namn unikt? Nej, eftersom det kan finnas flera personer med samma namn. Är adress en kandidatnyckel? Nej, eftersom det kan finnas flera personer som bor på samma adress (syskon, familj). Är postnr en kandidatnyckel? Nej, eftersom det kan finnas flera personer som bor inom samma postnummerområde. Samma gäller för postort. Postort är ingen kandidatnyckel eftersom det kan finnas fler personer som bor på samma postort.

I tabellen Medlem är det alltså personnr som är det enda attributet som är en kandidatnyckel. 3NF sa att det inte fick finnas några pilar som gick mellan attribut utanför någon kandidatnyckel. I tabell har vi en pil som går från postnr till postort (ffb2: postnr --> postort). Eftersom ingen av dessa attribut är en kandidatnyckel bryter tabellen mot 3NF. Tabellen Medlem uppfyller därför endast 2NF.

Om en tabell bryter mot någon av dessa normalformer (1-3) måste vi åtgärda detta eftersom vi annars kan få problem vid insättning eller borttagning av data. Med att åtgärda menar jag att vi måste flytta ut alla attribut som ingår i det determineringen som bryter mot aktuell normalform till en egen tabell. I den nya tabellen kommer determinanten att bli primärnyckeln. Determinanten lämnas även kvar i den ursprungliga tabellen, som där blir en främmandenyckel till den nya tabellens primärnyckel.

I vårt fall tar vi alla attribut i ffb2 (postnr och postort) och flyttar dessa till en ny tabell, som vi kan kalla för Postort. I denna tabell sätter vi determinanten (postnr) som primärnyckel. I den ursprungliga tabellen Medlem lämnar vi kvar postnr som en främmande nyckel till den nya tabellen. Vi får då följande tabeller:

Medlem (<u>personnr</u>, namn, adress, postnr*)
Postort (<u>postnr</u>, postort)

Boyce-Codd normalform (BCNF)

Slutligen säger BCNF att alla determinanter i en tabell måste vara en kandidatnyckel. Det får alltså bara finnas pilar som går från en kandidatnyckel och inte pilar som går till en kandidatnyckel. I vårt fall med de två tabellerna ovan är de enda determinanterna även kandidatnycklar, vilket innebär att Medlem och Postort även uppfyller kraven för BCNF.

Hoppas det blev lite klarare med hur vi går tillväga när vi normaliserar en tabell. I inlämningsuppgift 5 får du prova på att identifiera alla fullständigt funktionella beroenden (vilka attribut som determinerar vilka attribut) i olika tabeller och rita upp "pilar" så som jag har gjort i detta exempel. Därefter måste du identifiera alla kandidatnycklar för att senare kunna avgöra vilken normalform tabellerna befinner sig i. Bryter någon tabell mot 1NF-3NF måste du göra en uppdelning precis som jag beskrivit ovan.