Databasteori

Övningar

Eva L. Ragnemalm

November 2009, senast reviderad mars 2018

Observera: det finns inget facit till dessa övningar, eftersom många går att lösa på flera sätt, mer eller mindre olika. Om du är osäker på vilka kvaliteter din lösning har är du välkommen att kontakta mig!

Boken som refereras till i texten är kursboken, Databasteknik, av T. Padron-McCarthy och T. Risch, Studentlitteratur.

Innehåll:

ER-modellering

Skapa ER-diagram för nedanstående övningar (läs om ERmodeller i boken). Övningarna kräver inte EER-komponenter. Många övningar går att lösa på flera sätt (ibland kan man variera små detaljer, ibland radikalt olika strukturer). Fundera på vilka egenskaper de olika lösningssätten har (oftast får systemet olika begränsningar). För att bedömas som korrekt ska din lösning kunna representera allt som nämns i uppgiften.

1. Universitetet

(Detta är samma exempel som tas upp på föreläsningen om ER-diagram)

Antag att universitetet behöver ett databassystem för att hålla rätt på studenter som går program, vilka sektioner de är med i och vilka som betalt terminssavgiften.

Datakrav:

För att representera studenter behöver vi lagra namn, personnummer för att få ett unikt id, liu-id och lösenord, samt kontaktinformation: adress och telefonnummer. En adress består av gatans namn, husnummer, postnummer och postort. En student kan ha flera telefonnummer. En student är medlem i minst en sektion och antogs till ett program ett visst år. En student kan ha bytt program och har då antagits till ett annat program ett annat år (ett program per år).

De program som finns beskrivs med namn och en unik kod, ex f7ksa, och har en programansvarig (namnet lagras) och en studievägledare (namn). Ett program har en tillhörande studentförening, en sektion.

Sektionerna identifieras med namn och man lagrar också plats för sektionens lokal. En sektion kan organisera studenter från flera olika program (t.ex. både kandidatprogram och masterprogram i ett ämne). Sektionerna tar en terminsavgift för medlemskapet och man vill registrera den senaste terminen en student betalat för.

Funktionskrav:

Man vill kunna söka ut liu-id till de studenter som tillhör en viss sektion.

Man vill kunna hitta vilken sektion som organiserar vilka program, vilka studenter som går vilka program.

Man vill kunna se när (vilket år) studenten antogs till det/de program den är registrerad på.

Man vill kunna se vilken termin en student senast betalade sektionsavgiften.

Tillägg (mer avancerat)

Antag att sektionerna vill registrera sina fester, alltså de fester sektionen regelbundet arrangerar. Dessa identifieras med namn och har tidpunkt, budget och lokal. Eftersom sektionerna inte talar med varandra finns ingen garanti för att olika sektioner inte hittar på fester med samma namn. De måste då skiljas från varandra.

2. Företaget

Antag att ett företag med s.k. matrisorganisation (personalen är anställd på avdelningar, som har avdelningschefer, men arbetet organiseras i projekt där man plockar in folk från olika avdelningar efter behov) behöver hjälp att hantera information om sina anställda för lönehantering och arbetsplanering, samt att hantera de löneförmåner i form av familjeförsäkringar som de anställda har. Avdelningschefers lön är delvis beroende av hur länge de varit chefer för sina avdelningar.

Datakrav:

Företaget består av ett antal avdelningar. Varje avdelning har ett namn, ett nummer, en chef och ett antal anställda. Startdatum för varje avdelningschef registreras. En avdelning kan ha flera lokaler. Lokaler identifieras endast med sina namn.

Varje avdelning finansierar ett antal projekt. Varje projekt har ett namn, ett nummer och en lokal där man arbetar.

För varje anställd lagras följande information: namn, personnummer, adress, lön och kön.

En anställd jobbar för endast en avdelning men kan jobba med flera projekt som kan tillhöra olika avdelningar. Information om antalet timmar (per vecka) som en anställd planeras jobba med ett projekt lagras. Facket har drivit igenom ett krav på att ingen får beläggas mer än 40 timmar i veckan. Information gällande den anställdes avdelningschef behöver också finnas.

För varje anställd lagras information om familjen av försäkringsskäl. För varje familjemedlem lagras förnamn, födelsedatum, kön samt typ av relation till den anställde (make/maka resp barn). Man kan inte anta att informationen om familjemedlemmarna är unik. Man antar att anställda inte är gifta med varandra (vilket problem skulle kunna uppstå om vi inte antog detta).

Exempel på funktioner man sett behov av:

Lisa Ohlsson är sjuk idag, sök ut alla projekt där hon jobbar så att man kan skriva på lokalens whiteboard att hon är sjuk.

De anställda som har barn under 12 år ska få ett erbjudande om barnförsäkring (lista anställda med barn under 12)

Projekt X har drabbats av förseningar och behöver komma ikapp. Deltagarna behöver kunna lägga mer tid på projektet under den närmaste månaden. De avdelningschefer som har folk som jobbar på projekt X ska sammankallas för förhandlingar.

3. Varuhuset

Antag att ett större varuhus-företag med butiker över hela landet behöver hålla rätt på personalen och varorna, samt kunder som får hemleveranser. Man behöver hålla rätt på personalens löner och för varorna gäller det lagersaldo och leveranser från olika leverantörer (en viss vara kan levereras av olika leverantörer).

Datakrav

Varuhusföretaget har anställda, med namn och lön, som arbetar på varuhusets olika avdelningar (namn och nummer), där man säljer olika varor (namn och nummer). Varje avdelning har en chef, som är en av de anställda.

Varorna levereras av olika leverantörer (namn och adress), och flera leverantör kan leverera samma varor, men till olika priser, som också kan variera från gång till gång.

Varuhuset har hemkörningsservice. Kunder som har konto hos varuhuset och anmält en adress för leveranser kan beställa varor och få dem levererade hem. Varje sådan beställning har ett ordernummer och ett orderdatum, utöver listan av ingående varor (naturligtvis kan man beställa mer än en av varje vara vid ett visst tillfälle).

4. Sjukhusets förlossningsavdelning

Man dokumenterar förlossningar, varje förlossning involverar en mor och ett barn som föds (som inte kan identifieras med något personnummer ännu, utan identifieras med hjälp av modern och vilken tid förlossningen startade). En flerbarnsfödsel dokumenteras som olika förlossningar med olika start-tidpunkter (förlossningen för en yngre tvilling anses starta då den första tvillingen är ute). Personalen som medverkar är alltid en ansvarig barnmorska och en eller flera sköterskor. Det finns också alltid en förlossningsläkare till hands, men vid en normal förlossning blir denna aldrig involverad. All personal identifieras med anställnings-ID, och man lagrar namnet och vilken

avdelning man tillhör. För förlossningsläkare och sköterskor lagras specialistområde. Barnmorskor och läkare har personsökare, vars nummer man lagrar. Läkaren har också sin dagliga journal där denne skriver löpande anteckningar om alla ingripanden.

Man vill för varje förlossning veta vilken personal som var involverad och vilket barnets status på APGARskalan var. Förlossningsförloppet dokumenteras för varje förlossning (i ett textdokument), men om förlossningsläkaren ingriper skriver denne också sin egen dagliga journal för just den dagen.

Använd inte EER. Detta ER-diagram går att göra på två fundamentalt olika sätt. Utforska båda. Konvertera båda till relationsschema för att se om/vad det blir för skillnad på relationsschemat.

5. Mäklarfirman

Antag att en mäklarfirma behöver hjälp att hålla ordning på sina försäljningsobjekt, kunder och budgivning. Man vill också hålla ordning på de banker kunderna har kontakt med.

Datakrav

För varje objekt (fastighet) lagrar man adress, område, beskrivning, bild, boyta samt vilken typ av fastighet det är (lägenhet, villa, kedjehus). En viss mäklare (en av de anställda) är huvudansvarig för varje objekt. Varje objekt tilldelas en unik kod. Information om ägarna lagras också.

Varje mäklare har ett unikt ID, ett kontor och ett mobilnummer, och namn. Varje mäklare ansvarar för ett antal försäljningsobjekt.

Man lagrar också information om lånegivande banker/institut. För varje långivare (som har unika namn men också ges ett ID) har man en kontaktperson och mobilnummer till den personen, samt ett centralt telefonnumret till företaget. Man behöver ibland kontakta dessa långivare för att bekräfta bud.

Varje kund registreras med namn, adress och mobilnummer, samt hemtelefon och arbetstelefon. Man registrerar också alternativa kontaktpersoner (t.ex maka/make) med telefonnummer. En kund ges ett kundnummer för att kunna identifieras unikt.

Ett bud på ett objekt har ett visst belopp och läggs vid en viss tidpunkt, som markeras av en unik tidsstämpel för att säkert kunna visa i vilken ordning buden lagts på ett visst objekt. Ett bud på ett objekt kan inte vara lägre än ett tidigare bud. Ett bud läggs av en viss kund via en mäklare och stöds av en lånegivare. Ett bud måste förmedlas via en mäklare, men det behöver inte vara den mäklare som har huvudansvaret för försäljningsobjektet. Ett bud har oftast, men behöver inte ha, en stödjande lånegivare (ifall kunden inte behöver ta lån).

Exempel på funktioner(frågor):

Man vill kunna lista alla bud på ett visst objekt i tidsordning. Man vill kunna lista alla objekt en viss mäklare ansvarar för. Man vill kunna lista alla bud en viss lånegivare stött. Man vill lista alla kunder som använder sig av olika lånegivare för olika bud. Man vill kunna lista alla objekt en kund lagt bud på.

Relationsmodeller

Dessa övningar går ut på att konvertera ER-diagrammen till Relationsmodeller (se Kokboksreceptet - kap 6 - i boken eller föreläsningen om Relationsmodeller). I denna kurs ritar vi relationsscheman enligt nedanstående:

Relationsnamn/tabellnamn

Attributnamn	Attributnamn	Atributnamn	Referensattribut

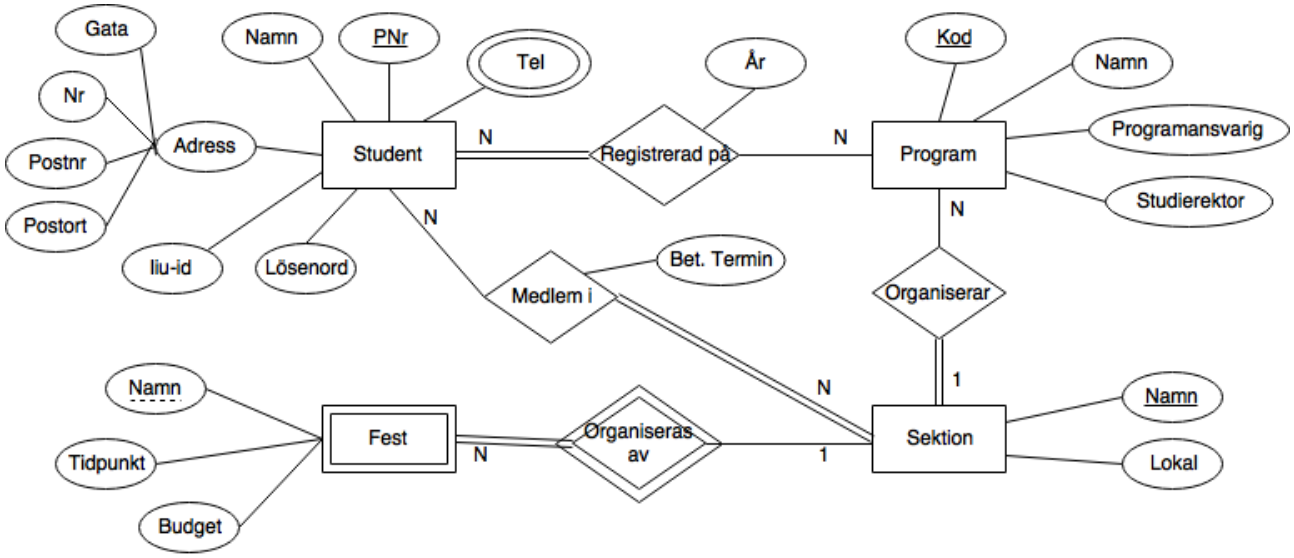
Relationsnamn/tabellnamn-1

Attributnamn	Attributnamn	Atributnamn	Attributnamn

Understrykning används för att identifiera primärnyckel (alla attribut som ingår i primärnyckeln stryks under) och pilar för främmande nycklar (se avsnittet Integritetsvillkor och Säkerhet). Observera att pilen är riktad från referensattributet till den relation där referensattributet utgör primärnyckel (för definition av begreppen, se boken eller ovannämnda föreläsning).

1. Universitetet

Se motsvarande övning i föregående kapitel för beskrivning och datakrav. Ett ER-diagram för det systemet kan se ut så här:



Semantiska integritetsvillkor: liu-id är unika. En instans av student kan ingå i flera sambandsinstanser av typen Registrerad på, men År får inte ha samma värde på dem (dvs en student kan bara antas till ett program per år och bara en gång per program). (notera att Studierektor borde vara Studievägledare).

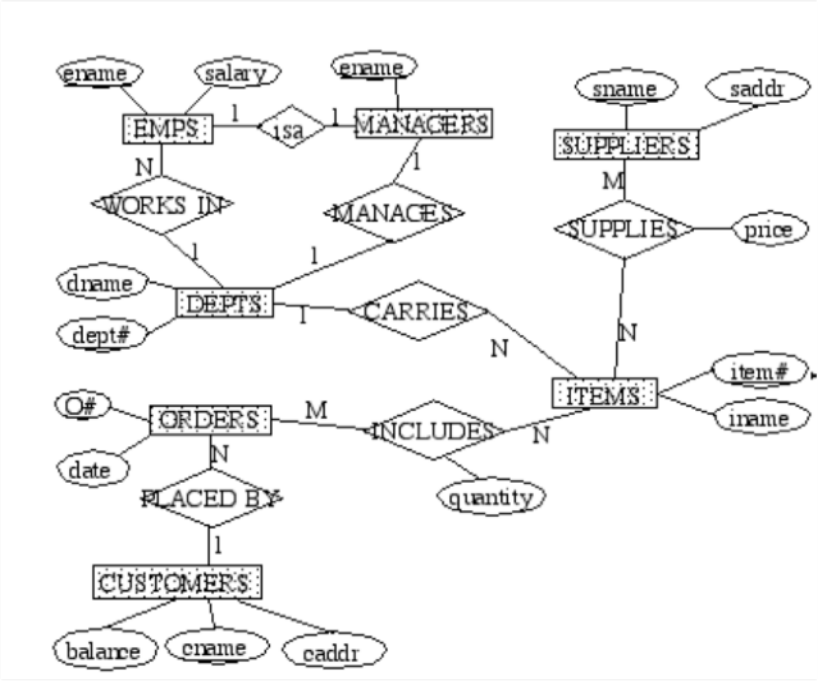
Konvertera till relationsschema.

2. Företaget

Använd det ER-diagram du kom fram till i övningen i föregående avsnitt och konvertera till relationsschema.

3. Varuhuset

Ett ER-diagram för övning 3 i föregående avsnitt kan se ut så här (andra varianter finns):



Konvertera det till relationsschema. Vissa förenklingar kan göras, diskutera vilka fördelar och nackdelar som finns med dem.

4. Sjukhuset

Konvertera bägge versionerna av ER-diagrammet i övning 4 i föregående avsnitt (flervägssamband respektive central entitetstyp med binära samband till övriga entitetstyper) till relationsschema om du inte redan gjort det. Diskutera skillnaderna.

5. Mäklarfirman

Konvertera det ER-diagram du skapade i övning 5 i föregående avsnitt till relationsschema.

Integritetsvillkor och säkerhet

Innehållet i en databas ska vara korrekt. Vilka tekniker finns för kontroller?

I den nedanstående texten används begreppet *relation* som i princip är samma sak som en tabell i en relationsdatabas och *attribut* vilket motsvarar en kolumn i en tabell. En *tupel* är en rad i en tabell. Ett *attributvärde* är innehållet i en cell i tabellen. Termerna relation/tupel/attribut används i första hand i teoretiska definitioner som kommer från relationsalgebra. Se avsnittet Normalisering för en fullständigare definition av begreppen.

1. Nycklar

I **ER-modellen** är en *nyckel* till/för en entitetstyp: **det eller de attribut som kan identifiera de unika entitetsinstanserna**. Den identifieras vid skapandet av entitetstypen på basis av den verklighet entitetstypen representerar. För varje värde på nyckeln kan det bara finnas en entitetsinstans, dvs nyckeln används för att unikt identifiera en viss entitetsinstans.

I **relationsmodellen** skiljer man på flera olika nycklar men gemensamt för dem är att de är **en delmängd av attributen i en relation som kan användas för att identifiera hela tupeln** i den relationen. Attributmängden som utgör en nyckel i en relation har alltid ett funktionellt beroende (se nedan) till alla attribut i relationen.

Relationsmodellen skiljer tre olika typer av nycklar: *supernyckel*, *kandidatnyckel* samt *primärnyckel*. En **Supernyckel** är en attributmängd vilken som helst som uppfyller ovanstående nyckelvillkor (inklusive mängden av alla attribut i relationen, eftersom hela tupeln alltid kan användas för att hitta sig själv). En **Kandidatnyckel** är en minimal attributmängd som uppfyller ovanstående (dvs inget attribut går att ta bort ur attributmängden utan att tappa möjligheten att identifiera hela tupeln). **Primärnyckel** är den kandidatnyckel i en relation som väljs av databasadministratören för att använda som referensattribut (se nedan) i andra relationer och ofta att sortera efter och indexera på. Enligt relationsmodellen får en tupel inte sakna värde på primärnyckeln.

Man hittar kandidatnycklar i en relation genom att ställa frågan: Vad representerar en tupel? Hur identifierar man det? (och finns det fler sätt att identifiera det?)

Liksom i ER-modellen används nyckeln i relationsmodellen för att unikt identifiera en viss tupel i en relation. Men den används också för att garantera att inga dubletter lagras.

I SQL används kommandot: `constraint primary key (attributnamn)` för att få den automatiska kontrollen av unik-heten och `not null` för att den inte kan vara tom.

2. Referensintegritet

Antag att databasen innehåller de två tabellerna Kurs och Anställd nedan. Tabellen Kurs representerar kurser som ges, tabellen Anställd de personer som ansvarar för kurserna. Attributet Ansvarig i tabellen Kurs innehåller anställningsnumret för den lärare som ansvarar för den kursen. Detta sätt att lagra information gör att vi inte behöver lagra all information om den ansvarige i kurstabellen, vilket skulle ställa till problem med dubbellagring, men vi kan fortfarande hitta informationen genom att följa länken i Ansvarig.

Kurs

Kursnamn	Kurskod	Poäng	Ansvarig

Anställd

AnstNr	Namn	Institution	Telefon

Attributet Ansvarig i tabellen Kurs kallas då **Referensattribut**, vilket är samma sak som främmande nyckel. Referensattribut illustreras i ett relationsschema som en pil som startar i

referensattributet och slutar i den utpekade relationen, se ovan. Om referensattributet är sammansatt av flera attribut (om den utpekade relationen har en sammansatt primärnyckel) har pilen flera ”bakändar”.

Referensintegritet innebär att när information på detta sätt delas upp i flera tabeller/relationer med koppling emellan så måste information som refereras till från den ena faktiskt finnas i den andra. Den lärare som pekas ut som ansvarig i Kurs-tabellen måste faktiskt finnas (i databasen) för att databasens innehåll ska vara korrekt. Tekniskt sett innebär detta att om det på en viss rad i Kurs står 123 i kolumnen ansvarig, så måste det finnas en rad i Anställd-tabellen som har 123 som värde i AnstNr. (Notera att det finns ingen begränsning åt andra hållet, ingenting kräver att ett visst värde i Anställd(AnstNr) måste finnas i Kurs(Ansvarig)). Kontroll av referensintegritet görs bland annat då data läggs till och tas bort.

I SQL definieras referensattribut med kommandot `constraint … foreign key` (som ingår i ett create table eller alter table kommando).

T.ex: `alter table Kurs add constraint fk-kurs-anstalld foreign key (kursansvarig) references Anstalld (anstnr);`

Not: För att få MySql att automatiskt utföra referensintegritetskontrollen måste tabellens "Engine" vara ”InnoDB” (Gäller mysql versioner 5.x). Andra databashanterare kan fungera annorlunda.

3. SQL injektion

Vad kan hända om man har ett formulär på nätet där man hämtar indata till en sql-sats? Antag att vi har en databas där vi lagrar personers namn och telefonnummer i en tabell med namnet Abonnent enligt nedan:

Abonnent

Namn	Telefonnr	hemligtNummer

Antag att vissa abonnenter har hemligt telefonnummer, dessa har värdet ”true” på hemligtNummer, övriga har värdet ”false”. Antag vidare att vi lägger upp ett formulär på nätet för att söka i denna tabell, men att de med hemligt nummer då inte ska presenteras. Vi har ett inmatningsfält i formuläret som levererar en sträng, som vi sedan sätter ihop med resten av ett fördefinierat SQL-uttryck och skickas till databashanteraren:

```
select namn, adress, telefonnummer from abonnent
where namn='$NAMN' and hemligtNummer=false;
```

Där $NAMN ersätts med den inmatade strängen. Om någon vill ställa till problem matar användaren in strängen ”Olle Karlsson' or 'a'='a' or ’a'='a"

sista raden i SQL-satsen blir då:

```
where namn= 'Olle Karlsson' or 'a'='a' or 'a'='a' and
hemligtNummer=false;
```

Den är alltid sann, och kommer att lista hela tabellen oavsett värde på attributet "hemligtNummer".

Läs mer i boken (kapitel 13.7) och googla också ”Little Bobby Tables” (XKCD är orginalet).

Normalisering

Normalisering är ett sätt att se till att en tabell (i relationsalgebra-termer en relation) inte innehåller redundant data, data som lagras på flera ställen i relationen. Man ser till att information inte dubbellagras i en relation genom att undersöka beroenden mellan attributen i en relation för att se om de uppfyller de s.k. Normaliseringsvillkoren. Det finns flera nivåer av normaliseringsvillkor (ibland kallade normaliseringsgrader eller normalformer), de mest använda är: 1:a normalformen (1NF), 2:a normalformen (2NF), 3:e normalformen (3NF) samt Boyce-Codds Normalform (BCNF). Det finns fler men de används sällan i praktiken eftersom det inte är givande.

De olika normalformerna beror av varandra i ordning, varför man, när man genomför normalisering alltid kontrollerar en relation mot de olika normalformerna i den ovanstående ordningen, alltså först 1NF, sedan 2NF, sedan 3NF till sist BCNF.

För risker och nackdelar med relationer som inte uppfyller BCNF, se boken (kap 11).

1. Normalisering som metod

När vi normaliserar i denna kurs går vi igenom relationerna (tabellerna) i databasen en och en, och för varje relation kontrollerar vi att relationen uppfyller de olika normaliseringsvillkoren (i den ovan nämnda ordningen). Om relationen inte gör det delar man upp den i två relationer genom s.k. informationsbevarande relationsschemauppdelning (se nedan). För varje uppdelning normaliserar man de två nya relationerna (oftast uppfyller de samma villkor som den ursprungliga).

Processen är alltså att vi tar en relation och kontrollerar om den uppfyller 1NF:

Villkoret för 1:a normalform är att:

alla attribut i relationen/tabellen är atomära, dvs odelbara.

När man använder ordet relation syftar man man på det matematiska begreppet relation, som definieras som ”en delmängd av kryssprodukten av ett antal domäner”, där en domän är en mängd attributvärden. Dessa attributvärden är per definition atomära. Alltså är attributen i en relation alltid atomära (läs mer om relationsalgebra i boken). När man talar om en tabell är frågan lite mer oklar, och man får undersöka om det finns anledning att tro att olika attribut eventuellt kan delas upp i flera i det specifika fallet. Ett attribut kan vara atomärt även om det består av en sammansatt datatyp som t.ex. en textsträng eller datumformat, det är hanteringen av det som avgör om det ska betraktas som atomärt. Om en sträng manipuleras i delar, t.ex. om den innehåller både gatunamn, postadress och postort och det görs sökningar efter geografiska områden mha postort är attributet formellt inte atomärt. Om det bara används i sin helhet för att skriva ut adresslistor eller adressera kuvert är attributet däremot atomärt.

Om en relation inte har atomära attribut är det ingen mening med att undersöka ytterligare nivåer av normalformer, de kan inte uppfyllas.

2. Funktionella beroenden, nyckelattribut och determinant

Om relationen uppfyller 1NF kan vi gå vidare och undersöka om den uppfyller högre normalformer. Definitionen av de högre normalformerna använder några begrepp som först måste introduceras: funktionellt beroende, nyckelattribut och determinant:

Funktionellt beroende

Givet att X och Y är delmängder av attributen i en relation R är definitionen av ett **Funktionellt Beroende (fb)** X=>Y (utläses ”ett funktionellt beroende från X till Y”, eller ”Y beror av X”) att **X bestämmer värdet på Y**. Det betyder att ett visst värde på X alltid kombineras med ett visst värde på Y (men inte omvänt). Detta ska gälla i alla möjliga förekomster av databasinstanser, dvs det bestäms av vad relationen representerar, vad som faktiskt kan förekomma.

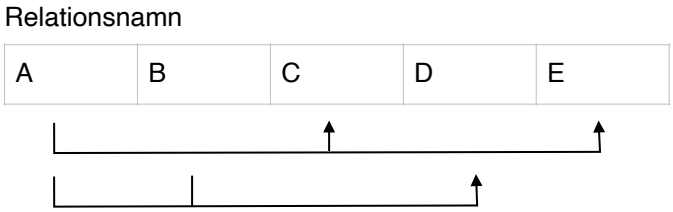
Ett exempel på funktionellt beroende är att i en relation som representerar personer, som innehåller namn och personnummer, finns det ett funktionellt beroende från personnummer till namn, alltså Pnr=>Namn. Namn beror av personnummer, men notera att det omvända inte gäller.

X, det eller de attribut som bestämmer, kallas **determinant** i beroendet. Ett trivialt funktionellt beroende existerar alltid från ett attribut till sig själv, och det är tillåtet att ha med ”onödiga” attribut i determinanten. Till exempel kan man säga att mängden av personnummer och namn bestämmer namn, alltså {personnummer, namn}=>namn.

Ett **Fullt Funktionellt Beroende (ffb)** X=>Y är ett funktionellt beroende sådant att man kan inte ta bort något attribut ur determinanten (X) och fortfarande ha ett funktionellt beroende.

Ett funktionellt beroende skrivs formellt i skrift som {Determinant} =>{bestämda attribut} men när den ena mängden består av ett enda attribut kan mängdklamrarna utelämnas. Notera att när mängden till vänster om pilen består av flera attribut betyder det att determinanten i det funktionella beroendet är en mängd, dvs att alla attribut i mängden bestämmer attributen till höger. När mängden på högersidan om pilen består av flera attribut betyder det dock att de olika attributen till höger var för sig bestäms av determinanten.

Ett funktionellt beroende kan också illustreras visuellt i ett relationsschema, med hjälp av en pil från determinanten till det/de bestämda attributen i relationen, se nedan:



En relation med attributen A..E har funktionellt beroende A=>{C,E}, samt funktionellt beroende {A,B}=>D. Lägg märke till var pilarna börjar och slutar.

Funktionella beroenden kan beräknas utifrån varandra. Om vi har en relation med attributen A, B och C och vet att vi har fb A=>B och B=>C så gäller också att A=>C. Detta kallas ett transitivt beroende.

Man hittar de fulla funktionella beroendena i en relation genom att systematiskt gå igenom den attribut för attribut och ställa frågan: vilket eller vilka andra attribut bestämmer värdet på detta attribut? Det är vad relationen faktiskt representerar som avgör svaret.

Funktionellt beroende och nycklar, nyckelattribut

Om det finns ett funktionellt beroende X=>Y där Y är alla attribut i relationen, är X en **supernyckel** i relationen (se Nycklar ovan). Om det finns ett fullt funktionellt beroende X=>Y där Y är alla attribut i relationen är X en **kandidatnyckel**. Mängden av alla attribut i relationen är alltid en supernyckel för relationen (dvs det finns alltid minst en nyckel: hela relationen).

De attribut i en relation som ingår i någon kandidatnyckel för relationen kallas för **nyckelattribut**. De attribut som inte ingår i någon kandidatnyckel kallas **icke-nyckelattribut**. Nyckelattribut markeras i relationen med asterisk (*) efter attributnamnet, men bara i samband med att man gör normalisering, inte i vanliga fall.

Det är viktigt att innan man fortsätter normaliseringen identifiera de viktigaste funktionella beroendena, samt alla kandidatnycklar i relationen.

För att undersöka om en relation uppfyller 2:a normalform kan vi nu formulera villkoret:

Villkoret för 2:a normalform:

relationen är i 1 NF plus att alla icke-nyckelattribut i relationen ska vara fullt funktionellt beroende av alla relationens kandidatnycklar.

Detta går också att formulera: det får inte finnas icke-nyckelattribut i relationen som är funktionellt beroende av en del av en kandidatnyckel.

Det betyder bland annat att om relationen bara har enkla kandidatnycklar, dvs kandidatnycklar som består av enstaka attribut, så uppfyller den automatiskt 2NF. Notera också att här kollar man INTE efter attribut som är beroende på annat än kandidatnycklar, utan bara att de behöver hela kandidatnyckeln för att bestämmas. En relation som inte uppfyller detta villkor måste delas upp i två, se Informationsbevarande relationsschemauppdelning, nedan.

Om relationen uppfyller villkoren kan vi gå vidare och undersöka om den uppfyller 3NF:

Villkoret för 3:e normalform:

relationen är i 2 NF plus att för varje fullt funktionellt beroende X=>Y i relationen måste Y vara primattribut eller determinanten en supernyckel.

Det går också formulera; inget icke-nyckelattribut får vara fullt funktionellt beroende av något annat icke-nyckelattribut.

Om vår relation inte uppfyller det måste den delas i två, se informationsbevarande relationsschemauppdelning, nedan. Om den uppfyller villkoret för 3NF kan vi gå vidare och undersöka om den uppfyller Boyce-Codds normalform:

Villkoret för Boyce-Codds normalform (BCNF):

3NF plus att för varje fullt funktionellt beroende X=>Y i relationen måste X vara supernyckel.

Det går också formulera: inget attribut får vara ffb av något icke-nyckelattribut, eller förenklat: 1NF plus att det inte får finnas några ffb utom de som utgår från kandidatnycklarna.

Om relationen inte uppfyller villkoret och man vill uppfylla BCNF måste den delas i två. Det finns tillfällen då man inte nödvändigtvis vill uppfylla BCNF för att det är svårt att dela relationen utan att bryta funktionella beroenden. Det kan innebära att viss data går förlorad, och då kan det i vissa fall vara värt att behålla den grad av dubbellagring som 3NF medger.

3. Informationsbevarande relationsschemauppdelning

När man har redundans i sin relation (dvs bryter mot någon normalform) har man alltså någon form av dubbellagrad information. Då måste man bryta ut den information som dubbel-lagras. Det gör man genom att skapa två relationer av den ursprungliga relationen. Uppdelningen ska vara **informationsbevarande**, dvs det ska gå att återskapa den ursprungliga relationen (exakt) genom att göra en naturlig join[1] mellan de två nya. Man får inte bryta något fullt funktionellt beroende genom att sära på determinant och de attribut som bestäms av den i olika relationer.

Ett tips på hur man kan tänka här är att skapa en ny relation som representerar det funktionella beroende som gör att relationen bryter mot normalformen i fråga. Den nya relationen ska då innehålla determinanten och de attribut som bestäms av den. I den ”gamla relationen” låter man determinanten till det aktuella funktionella beroendet vara kvar och stryker de beroende attributen. Då kan man alltid återskapa data genom att determinanten blir främmande nyckel i den ”gamla relationen”.

Den ”gamla relationen” bör behålla namnet från ursprungsrelationen medan den nya bör ges ett namn som beskriver vad den representerar.

Ett problem uppstår om något av de beroende attributen (som alltså ska strykas i den ”gamla relationen”) är eller ingår i en determinant i ett annat fullt funktionellt beroende än det aktuella. Det finns då ett transitivt funktionellt beroende som man ska beräkna och använda istället, dvs man lyfter ut allt som bestäms av determinanten i fråga.

När man har delat upp en relation i två på detta sätt bör man iterativt normalisera dem.

---

[1] Läs om relationsalgebra-operationen join i boken (kap. 10.11-12)

Normaliseringsövningar

1. Universitetet

Ett relationsschema för databasen från övning 1 i kapitel 1 och 2 är detta (du kan ha gjort andra val i kapitel 2 som är lika korrekta eller bättre):

Student (PNr, Namn, Gata, GatNr, PostNr, PostOrt, liu-id, Lösen)

StudentTel (Student, Tel)

Program (Kod, Namn, Programansv, Studievägledare, Sektion)

Sektion (Namn, Lokal)

Fest(Sektion, Namn, tidpunkt, budget)

Reg-På (Student, Program, År)

Medlem-I (Student, Sektion, BetTermin)

Antag att vi kommer på att vem som är programansvarig och vem som är studievägledare för ett program kan variera mellan olika år och vi behöver kunna lagra personerna som hade dessa roller respektive år. Vi måste då lägga till ett attribut År i relationen Program och definiera nyckeln som sammansättningen av Kod och År. Detta medför att vi får en entitetsinstans av Program för varje år, vilket var vad vi behövde för att kunna lagra olika namn på olika år. Denna ändring betyder också att vi kan ta bort attributet År på sambandet Registrerad-På mellan Student och Program, eftersom en student registreras på en program-instans som nu redan har ett visst år. Detta medför i sin tur att relationen som representerar Registrerad-På (som vi döpte till Reg-På) fortfarande har med attributet År, men att det nu ingår i nyckeln till relationen (hur detta går till är en bra övning!).

Program och Reg-På får då följande schema:

Program (Kod, År, Namn, Programansv, Studievägledare, Sektion)

Reg-på (PgmKod, År, Student)

I övrigt är schemat oförändrat.

Innehåll i Program skulle då kunna vara:

Kod	År	Namn	Programansv	Studievägledare	Sektion
LIU500007	2019	Kognitionsvetenskapliga programmet	Mattias Arvola	Katarina Isotalo	KogVet
LIU500007	2020	Kognitionsvetenskapliga programmet	Erik Prytz	Katarina Isotalo	KogVet
LiU50009	2020	Statistik och dataanalys	Linda Wänström	Isak Hietala	StatLin
LiU50009	2019	Statistik och dataanalys	Linda Wänström	Isak Hietala	StatLin

Notera att ett programs namn och sektion är oförändrad år från år.

Normalisera Kurs (se ovan).

2. Biblioteket

Linköpings oberoende bibliotek har en databas som håller reda på alla deras böcker. De är inte så många, men de är väldigt oberoende. Dessutom innehåller den registrerade låntagare och aktuella lån. Tabellerna, med innehåll, finns nedan. Tabellinnehållet kan användas för att motbevisa teorier om funktionella beroenden (men inte bevisa).

Dock är bibliotekets databasdesign kass. Analysera den existerande databasen, förklara vilka problem den har och föreslå en bättre design. (Dvs normalisera respektive tabell). Om du vill kan du därefter göra en helt ny design genom att börja från början med ER-diagram, konvertera till relationsschema och normalisera. Blev det någon skillnad?)

Det finns tre tabeller:

- En tabell som heter BOOK som innehåller data om bibliotekets böcker. Den har attributen TitleNr vilket är ett löpnummer som biblioteket ger till varje bok, ISBN (olika utgåvor, olika år, av en bok har olika ISBN, och får också olika TitleNr), CopyNr som används för att skilja på olika exemplar av samma bok, Title, PublYear, Author (man lägger in en rad (tupel) per författare för böcker med flera författare - varje rad har alltså bara en författare), AuthorNat samt Condition vilket är bokexemplarets (fysiska) skick. Primärnyckel är satt till kombinationen av TitleNr, CopyNr och Author.
- En tabell som heter CUSTOMER, och som innehåller data om låntagarna. Den har attributen CustomerNr som är ett unikt nummer som biblioteket ger varje låntagare, PersonNr som är personens personnummer, Name, Address och Tel. CustomerNr är primärnyckeln.
- En tabell som heter LOAN, där information om lån är lagrad. Den har attributen TitleNr, CustomerNr, CopyNr, Date vilket är datumet då boken lånades ut, samt BorrowerName som är låntagarens namn. Primärnyckel är kombinationen av TitleNr och CopyNr.

Databasinnehållet

Tabellerna ser för närvarande ut såhär:

BOOK

TitleNr	ISBN	CopyNr	Title	PubYear	Author	AuthNat	Cond
1	7114810	1	Database system concepts	1997	A. Silberschatz	USA	Good
1	7114810	1	Database system concepts	1997	Henry F. Korth	USA	Good
1	7114810	1	Database system concepts	1997	S. Sudarshan	India	Good
2	805317538	1	Fundamentals of database systems	1994	Ramaz Elmasri	USA	Mint
2	805317538	1	Fundamentals of database systems	1994	S. B. Navathe	USA	Mint
2	805317538	2	Fundamentals of database systems	1994	Ramaz Elmasri	USA	Prist. Mint
2	805317538	2	Fundamentals of database systems	1994	S. B. Navathe	USA	Prist. Mint
3	198642253	1	Mord	1996	Jan Guillou	Sweden	Poor
3	198642253	2	Mord	1996	Jan Guillou	Sweden	Good
4	3411021764	1	Våld	1998	Jan Guillou	Sweden	Poor

CUSTOMER

1. CustomerNr	1. PersonNr	1. Name	1. Address	1. TelNr
1	6312111658	Thomas Padron-McCarthy	Vägen 7	282627
2	4403149001	Lena Strömbäck	Gatan 6	123456
3	5901239023	Eva Ragnemalm	Vägen 3	987654
4	5612067832	Arnold Scwarznegger	Stigen 8	345634

LOAN

1. TitleNr	1. CopyNr	1. CustomerNr	1. Date	1. BorrowerName
1	1	3	23 nov. 2009	Eva Ragnemalm
3	2	1	1 nov. 2009	Thomas Padron-McCarthy
2	1	2	1 dec. 2009	Lena Strömbäck

OBS: om man följer ”algoritmen” att undersöka normalformerna i tur och ordning och gör informationsbevarande relationsschemauppdelningar på korrekt sätt kommer man att ha en konstig Book-tabell kvar på slutet som man inte blir av med. Då har man gjort rätt... Det är inte alltid man kan rädda en dålig design genom att normalisera på detta sätt. Metoden förutsätter att de kandidatnyckelberoenden som finns är relevanta.

3. Andrahandsuthyrningsfirma

En firma som administrerar andrahandsuthyrning av lägenheter vill hålla reda på kontraktisinformationen.

Man vill hålla reda på vem hyr vad (kundnamn, kundnummer, lägenhetsnummer, lägenhetsadress) när (start och slutdatum) samt till vilken hyra (som är olika för varje lägenhet). Man vill att gamla kontrakt ska gå att hitta, men om samma person hyr samma lägenhet en gång till ersätts detta kontrakt med det nya. En lägenhet kan bara hyras av en person åt gången.

De lagrar också information om vem som egentligen äger lägenheten.

För ”tills vidare”-kontrakt registreras slutdatum som null.

Varje person antas bara hyra varje lägenhet en gång och kan bara hyra en lägenhet åt gången. En ägare kan dock låta firman hantera flera lägenheter.

Möjligt (icke-optimalt) relationsschema:

Kontrakt(kundNr, lghNr, kNamn, lghAdr, start, slut, hyra, ägarNr, äNamn)

KundNr	Lgh Nr	Kund Namn	Lgh Adr	Start	Slut	Hyra	Ägar Nr	Ägar Namn
CR76	PG4	J.Kay	Lagv 4	81112	90312	3200	CO40	T.Moe
CR76	PG16	J.Kay	Nyv 12	90313	null	3500	CO93	U.Sin
CR56	PG4	A.Son	Lagv 4	90313	null	3200	CO40	T.Moe
CR56	PG35	A.Son	Husg 1	71112	90310	3000	CO93	U.Sin
CR56	PG16	A.Son	Nyv 12	60801	71110	3500	CO93	U.Sin

4. Inspektion

Antag att uthyrningsfirman inspekterar varje lägenhet mellan uthyrningarna och noterar brister och problem. När inspektion ska göras bokar inspektören en bil som används under dagens inspektioner. En bil kan bokas av två personer under samma dag eftersom inspektörerna ofta är ute enbart förmiddag eller eftermiddag, men en inspektör byter inte bil under dagen (kan alltså bara boka en bil per dag). En inspektör kan inspektera flera lägenheter under samma dag, men varje lägenhet inspekteras endast en gång en viss dag.

Det finns alltså ett (eller flera) rapportformulär som nedanstående per lägenhet, dvs information om lägenhetsnummer och lägenhetsadress ska också lagras i databasen:

Lägenhetsnummer: PG4
Lägenhetsadress Studentv 8 Nollköping

Inspektionsdatum	Tid	Kommentar	InspektörNr	Inspektör Namn	BilNr
90228	10.00	Trasigt proslin	SG37	Ann Beech	ABC123
81130	09.00	Fint	SG14	David Ford	DEF098
71130	13.00	Mögel i badrum	SG14	David Ford	GHI987

Normalisera relationen.

5. Bilar

Övning ur kursboken, sid 234.

Man vill representera ett antal medlemmar i en förening (bilklubb) och deras bilar. Personerna har medlemsnummer (unikt) och namn samt ett antal bilar var. Man kan tänka sig att representera detta på följande olika sätt:

Medlem 1

Nummer	Namn	Bilar
1	Olle	RFN540
2	Stina	null
3	Saddam	SQL123, DBA456, WCA912

Medlem 2

Nummer	Namn	Bil 1	Bil 2	Bil 3
1	Olle	RFN540	null	null
2	Stina	null	null	null

Nummer	Namn	Bil 1	Bil 2	Bil 3
3	Saddam	SQL123	DBA456	WCA912

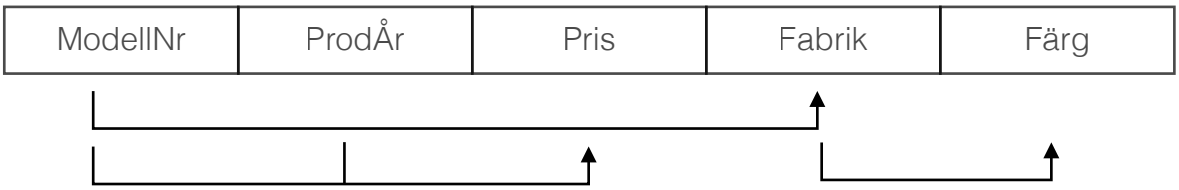
Medlem 3

Nummer	Namn	Bil
1	Olle	RFN540
3	Saddam	SQL123
3	Saddam	DBA456
3	Saddam	WCA912

Vilka problem finns med dessa lösningar? Finns det något bättre sätt? Och vad har detta med normalformer att göra?

6. Kylskåp

Antag att du har en relation Kylskåp enligt nedan, med de fulla funktionella beroenden som finns utritade (de skrivs: ModellNr=>Fabrik, ModellNr, ProdÅr=>Pris, Fabrik=>Färg)



Fungerar följande attribut/attributmängder som kandidatnyckel? Motivera.

a) {ModellNr}

b) {ModellNr, ProdÅr}

c) {ModellNr, Färg}

Baserat på den/de kandidatnycklar du identifierar, ange vilken normalform Kylskåp uppfyller och varför.