DD1370: Databaser och Informationssystem **Hösten 2014**

Del 2: ER-modellering och överföring till Databasstruktur v0.9

Petter Ögren *1:e December*

Disclaimer: *Dessa anteckningar har producerats under viss tidspress, och kan därför innehålla smärre fel och oklarheter.*

Sammanfattning Detta dokument beskriver mycket kortfattat de viktigaste delarna av ER-modellering och överföringen av ER-modeller till databasstrukturer. ER står för Entity-Relationship och är ett grafiskt sätt att modellera en problemdomän. Erfarenhet visar att det ofta lönar sig att först göra en ER-modell, och sedan bestämma vilka tabeller som skall finnas i databasen, den så kallade databasstrukturen.

En ER-modell består av ett antal pusselbitar, Entiteter, Attribut och Samband, samt olika varianter på dessa pusselbitar. I detta dokument skall vi beskriva hur dessa varianter ser ut och när de används. När man sedan skapar en databas baserat på ER-modellen, kommer de olika varianterna av pusselbitarna att ge upphov till olika utseende på tabellerna i databasen.

1 Entiteter och Attribut

ER i ER-modell står för Entity-Relationship, eller Entitet-Samband på svenska. Entiteter är objekt, och Sambanden är just samband mellan sådana objekt. I detta stycke skall vi fokusera på Entiteterna, och deras egenskaper, så kallade Attribut. Sambanden återkommer vi till i nästa stycke.

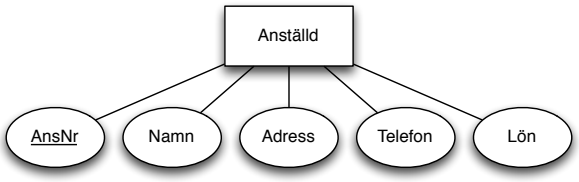
Som vi sett tidigare består en databas av en samling tabeller, där varje tabell har ett antal kolumner. Ofta, men inte alltid (se nedan), motsvaras tabellerna av Entiteter, och kolumnerna av Attribut. Om vi t.ex. har följande tabell

Anställd				
AnsNr	Namn	Adress	Telefon	Lön
1	Kalle	Vintervägen 1	070-111111	20 000
2	Anna	Sommargatan 2	070-222222	40 000
3	Kim	Sommargatan 7	070-343434	40 000
⋮	⋮	⋮	⋮	⋮

så motsvaras den av ER-diagrammet i Figur 1.1. Notera hur Entiteten ritas som en rektangel, och Attributen som ovaler. Entiteten (dvs Objektet) är en anställd, och Attributen (egenskaperna) är anställningsnummer, namn, adress, lön etc.

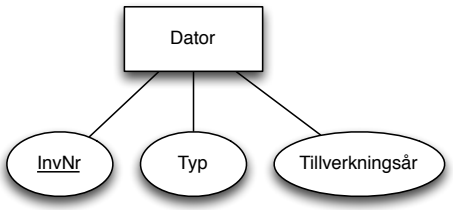
Om vi istället har följande tabell

Dator		
InvNr	Typ	Tillverkningsår
1	Macbook	2013
2	Dell PC	2009
⋮	⋮	⋮



Figur 1.1: En entitet *Anställd* med attributen *AnsNr, Namn, Adress, Lön* och *Telefon*.

så motsvaras den av ER-diagrammet i Figur 1.2.



Figur 1.2: En entitet *Dator* med attributen *InvNr, Typ* och *Tillverkningsår*.

Entiteten (dvs Objektet) är en dator, och Attributen (egenskaperna) är InvNr Typ, Tillverkningsår (InvNr är en förkortning av Inventarienummer, ett unikt nummer i företagets lista över saker).

Alla entiteter måste ha ett eller flera attribut som är så kallade *nyckelattribut*. Nyckelattributen är något som unikt identifierar Entiteten. I både tabeller och ER-modeller stryker vi under nyckelattributen. I exemplen ovan är alltså AnsNr nyckelattribut till Anställd, och InvNr är nyckelattribut till Dator. Ibland kallas nyckelattribut även för primärnyckel. I detta dokument kommer vi att använda begreppen som synonymer, och tillämpa dem på både ER-modeller och tabeller[1].

Vi kommer att återkomma till Entiteter och Attribut, men först skall vi beskriva Samband (Relationship, R i ER-modellen).

2 Samband mellan Entiteter

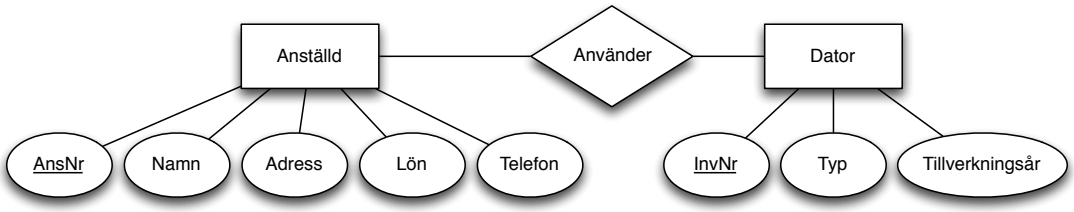
Ett Samband visar hur två eller flera Entiteter hänger ihop. De två entiteterna i exemplen ovan, Anställda och Datorer, hänger till exempel ihop genom att en Anställd *Använder* en Dator. Detta visar vi i ER modellen genom att rita en romb (eller diamant) som kopplar ihop de två Entiteterna så som i Figur 1.3.

Vi kan också välja att rita ER-modellen utan Attributen. Då ser den ut som i Figur 1.4.

2.1 Olika sorters 2-vägssamband

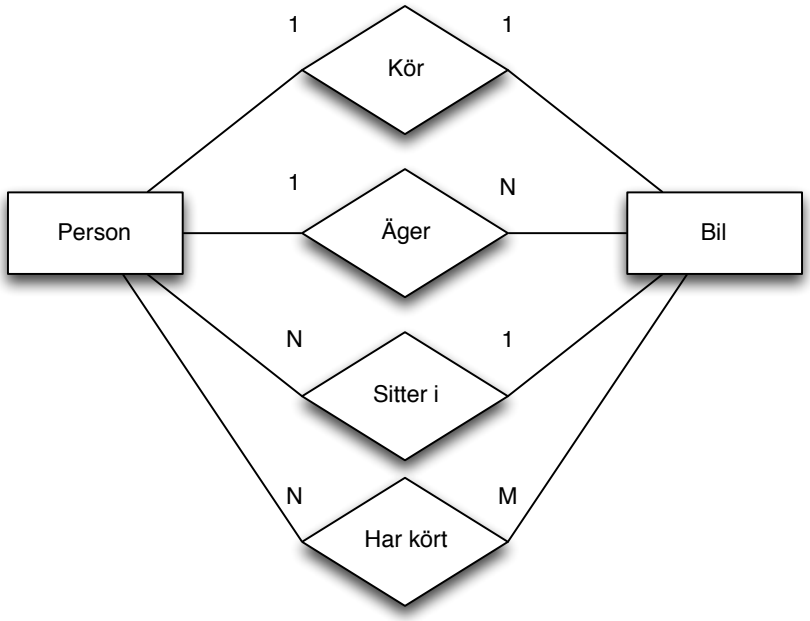
Vi skall nu diskutera olika sorters samband, och då använder vi Entiteterna Person och Bil som exempel. En person och en Bil kan hänga ihop på många olika sätt. Några av dessa finns illustrerade i Figur 1.5.

[1] Det finns dock de som hävdar att det finns en nyansskillnad. Den nyfikne kan Googla begreppen.



Figur 1.3: Sambandet *Använder* kopplar ihop Entiteterna *Anställd* och *Dator*.

Figur 1.4: Utan attribut utritade ser Figur 1.3 ut såhär.



Figur 1.5: De fyra olika 2-vägssambanden: 1:1, 1:N, N:1 och N:M.

En person kan *köra* en bil, *äga* en bil, *sitta i* en bil och *ha kört* en bil.

Förutom att dessa samband innebär olika saker i praktiken är de dessutom olika vad gäller antalet deltagande Entiteter på varje sida av sambandet. Detta symboliseras av de små 1:orna och N:en i figuren.

Kör är ett så kallat 1:1-samband. 1 person kan bara styra 1 bil åt gången, och 1 bil kan bara styras av 1 person åt gången.

Äger är ett 1:N samband. 1 person kan äga en eller flera bilar, men en bil kan bara ägas av en person. Det

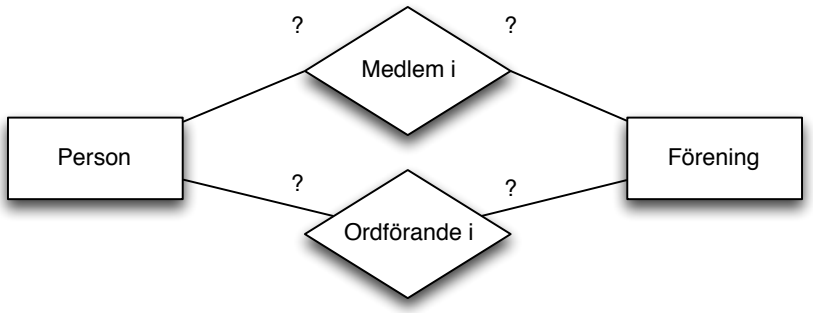
senare kan tyckas lite märkligt, men enligt svensk lag kan man inte samäga bilar.

Sitter i är ett N:1 samband (senare skall vi inte göra skillnad på 1:N och N:1 samband, men här noterar vi att det faktiskt är olika saker beroende på vilken sida N:et står). Flera personer kan sitta i samma bil, men en person kan inte sitta i flera bilar (samtidigt).

Har kört är slutligen ett N:M samband (man skulle kunna skriva N:N också, men skriver N:M för att understryka att det inte behöver vara lika många på respektive sida). Flera personer kan ha kört samma bil, och en person kan ha kört flera olika bilar.

2.2 Hur man avgör vilken sort ett samband är?

Givet ett samband, hur avgör man då vilken sort det är? Betrakta Figur 1.6.



Figur 1.6: Vilken typ är dessa 2-vägssamband?

När man skall avgöra vilken typ ett samband är kan man göra på följande vis

1. Kalla entiteterna V (vänster) och H (höger).
2. Föreställ dig först 2 exempel av varje entitet. Kalla dem V1, V2, H1 och H2.
3. Kan V1 uppfylla sambandet med både H1 och H2? Om svaret är Ja, så skall det stå ett N (eller M) till höger. Om svaret är Nej, så skall det stå en 1:a till höger.
4. Kan H1 uppfylla sambandet med både V1 och V2? Om svaret är Ja, så skall det stå ett N till vänster. Om svaret är Nej, så skall det stå en 1:a till vänster.

Nu följer vi receptet ovan. Vi börjar med sambandet *Medlem i*.

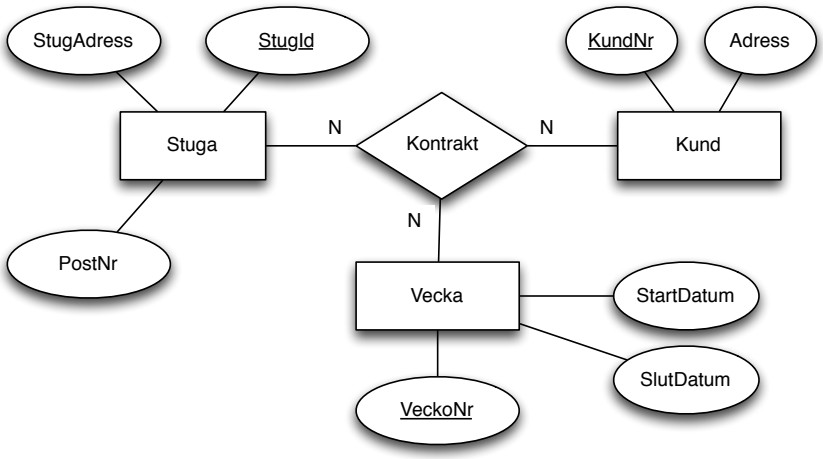
1. Person blir V och Förening blir H.
2. Lisa och Kalle blir V1 och V2. Djurgården och Hammarby blir H1 och H2.
3. Kan Lisa uppfylla sambandet (Medlem i) med både Djurgården och Hammarby? Ja, då skall det stå ett N (eller M) till höger.
4. Kan Djurgården uppfylla sambandet (Medlem i) med både Kalle och Lisa? Ja, alltså skall det stå ett N till vänster.

Således är Medlem ett N:M-samband.

Gör vi på samma sätt med *Ordförande i* så finner under punkt 3 att Lisa kan vara Ordförande i både Djurgården och Hammarby, vilket gör N till höger, men att Djurgården inte kan ha både Lisa och Kalle som Ordförande, vilket gör 1 till vänster. Alltså är Ordförande i, ett 1:N samband.

2.3 Flervägssamband

Ett samband kan också koppla ihop fler än två entiteter. Det kallas då Flervägssamband, se Figur 1.7.



Figur 1.7: Kontrakt är ett flervägssamband mellan en Kund, en Stuga och en Vecka, sambandet innebär att kunden kommer att hyra stugan under denna vecka.

Nu när vi har tittat på olika sorters samband kommer turen till olika sorters Attribut.

3 Olika sorters Attribut

Attribut kan också vara av olika sorter, vilket återigen påverkar hur tabellerna ser ut i databasen man skapar utifrån ER-modellen.

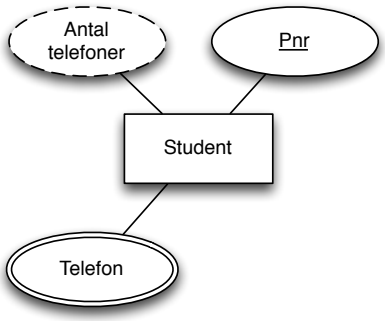
3.1 Nyckelattribut

Det viktigaste attributet är det unika *Nyckelattributet/Primärnyckeln*. Exempel på nyckelattribut är personnummer, kundnummer, registreringsnummer, telefonnummer, email-adress. Nyckelattributet visas i ER-modellen genom att namnet är understruket, se Pnr i Figur 1.8.

Det finns även något som heter *Partiell nyckel*. Den sitter på Svaga Entiteter (se avsnitt 4.5 nedan).

3.2 Flervärda attribut

Ibland vet man inte i förväg hur många 'objekt' en entitet kommer att ha av ett attribut. Till exempel telefonnummer. En person kanske har två telefonnummer (både hem och mobil), en annan kanske har en, och en tredje kanske saknar telefon. Istället för att skapa flera attribut (hemtelefon, jobbtelefon, mobiltelefon, etc) kan man då låta attributet vara flervärt. Det betyder att entiteten kan ha hur många som helst, eller inga. Det flervärda attributet ritas med dubbelsträck, som exemplet Telefon, i Figur 1.8.



Figur 1.8: Entiteten Student har ett Nyckelattribut Pnr, ett flervärt attribut, Telefon och ett härlett attribut Antal telefoner.

3.3 Härledda attribut

Ibland vill man lägga till ett attribut i modellen som inte innehåller någon egen data, utan räknas fram ifrån andra attribut. Till detta kommer vi att använda Vyer i SQL. Exempel på detta är *antal telefoner* (se Figur 1.8) antal kvadratmeter (som kan räknas ut från de olika rummens ytor) värde av en aktieportfölj (som kan räknas ut från gjorda transaktioner och aktuella kursdata). Härledda attribut ritas med en streckad linje, som i Figur 1.8.

4 Mer sofistikerad ER-modellering

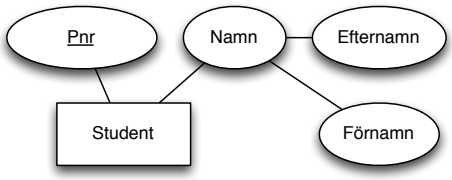
Nu skall vi titta på några exempel av lite mer avancerad ER-modellering.

4.1 Sammansatta attribut

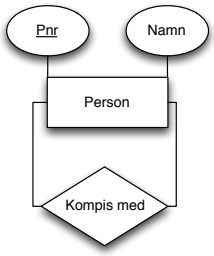
Ibland kan attribut utgöras av flera delar. T.ex. utgörs ett personnamn ofta av ett förnamn och ett efternamn, som i figur 1.9. Då ritar man helt enkelt de nya del-attributen sittande på det sammansatta attributet. På samma sätt kan en emailadress modelleras med ett sammansatt attribut, där `kim@kth.se` är sammansatt av ett användarnamn `kim`, ett domännamn `kth` och en toppdomän `se`.

4.2 Loopade samband

En Entitet kan mycket väl ha ett samband med sig själv, se Figur 1.10. Detta betyder dock inte nödvändigtvis att en person är kompis med sig själv, utan istället att en person är kompis med en (annan) person. Där



Figur 1.9: Attributet Namn är sammansatt av Förnamn och Efternamn.

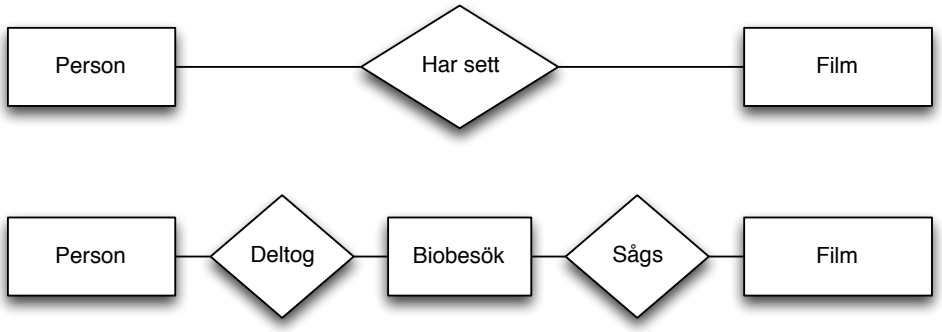


Figur 1.10: En Entitet kan ha ett samband med sig själv, vilket är praktiskt när man skall modellera samband mellan samma kategori av objekt. T.ex. en person är kompis med en annan person.

båda personerna representeras av samma Entitet.

4.3 Objektifiering av samband

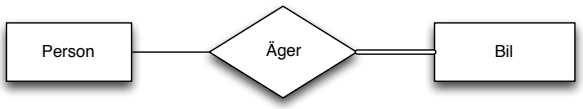
Ett samband identifieras genom de entiteter det sammanbinder. Detta medför att man bara kan spara ett tvåvägssamband mellan varje par av entiteter. Vill man istället kunna spara flera samband mellan varje par, så får man göra om sambandet till en entitet, och införa nya samband mellan de tre entiteterna. Ett exempel på detta finns i Figur 1.11. Här vill vi kunna spara flera biobesök, där samma person sett samma film flera gånger.



Figur 1.11: Sambandet *Har sett* objektifieras till Entiteten *Biobesök*, med de nya sambanden *Deltog* och *Sågs*.

4.4 Fullständigt deltagande

Ibland vill vi kräva att en entitet deltar i ett samband. Om t.ex. en person har ansvar för något. I Figur 1.12 har vi entiteterna Bil och Person och sambandet Äger. Om en bil är felparkerad måste ägaren betala böterna. Därför blir det problematiskt om en bil är ägarlös, men ändå körs omkring. Om vi kräver att alla entiteter deltar i ett samband, så ritar vi dubbelsträck på den sidan av sambandet där kravet finns. I exemplet kräver vi att varje bil ägs av någon, men vi kräver inte att varje person äger en bil. Därför är det enkelstreck på person-sidan av sambandet.

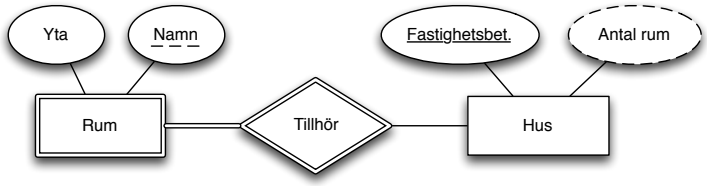


Figur 1.12: Fullständigt deltagande markeras med dubbelsträck. Varje bil måste ha en ägare.

4.5 Fullständigt deltagande, Svaga entitetstyper och Identifierande samband

Fullständigt deltagande beskrev vi ovan, men det kan också dyka upp om entiteten i sig inte kan *existera* utan sambandet och den andra entiteten. Om vi har en databas med lägenheter, och rum som tillhör lägenheterna, så kanske vi inte vill hålla reda på rummen separat. Istället för att numrera alla rum i ett flerfamiljshus och sedan låta lägenhet 21 innehålla kök 88, så vill vi nöja oss med att köket ifråga tillhör lägenhet 21.

Om en entitet inte kan existera utan en annan, kallar man den för en *Svag entitet*. Sambandet kallas sedan *Identifierande samband*, och deltagandet blir följaktligen ett *fullständigt* deltagande (som vi beskrev ovan). Alltihop ritas med dubbelsträck, se Figur 1.13. (Notera dock att dubbelsträckade attribut inte har med detta att göra, de ritas också med dubbelsträck, men är något helt annat.)



Figur 1.13: Rum är en svag entitetet, med partiell nyckel Namn, identifierande samband Tillhör och fullständigt deltagande i detta samband.

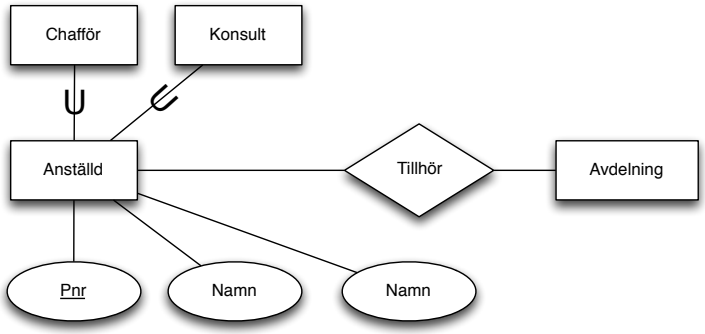
Eftersom den svaga entiteten inte kan existera utan den andra behöver den inget nyckelattribut. Istället har den en *Partiell nyckel*, som markeras med en streckad understrykning. I detta fall är Namn partiell nyckel. Som vi senare skall se kommer den partiella nyckeln i kombination med nyckalattributet för den andra entiteten att bilda primärnyckel i databasen, se avsnitt 5.3 nedan.

4.6 Arv

Ofta hanterar man i databaser olika objekt som är specialliseringar av andra objekt. T.ex. är en *Konsult* en speciell sorts *Anställd*, en *Aktie* en speciell sorts *Värdepapper*, en *Banan* en speciell sorts *Frukt*, en *VIP-kund* en speciell sorts *Kund*, osv. I dessa fall kan det vara praktiskt att både ha med den mer allmänna

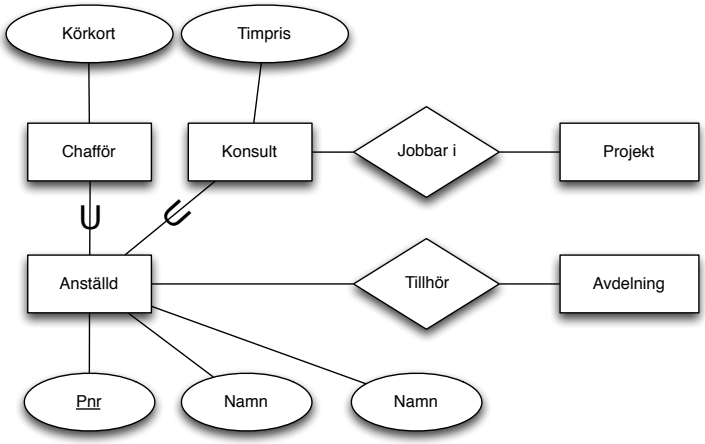
kategorin och den speciella i databasen, samt attribut och samband för dem båda. Med terminologi från programmeringsvärlden kallar vi den allmänna kategorin för superklass och den speciella för subklass, och säger att subklassen ärver egenskaper (attribut och samband) från superklassen.

För t.ex. Anställda skulle det kunna se ut som i Figur 1.14. Chaufför och Konsult är subklasser till superklassen Anställd. Både Chaufförer och Konsulter är anställda, tillhör avdelningar och har personnummer, namn och adresser angivna i databasen. Att dessa attribut och samband sitter på superklassen innebär att vi inte behöver upprepa dem för varje subklass.



Figur 1.14: Chaufför och Konsult är subklasser till superklassen Anställd.

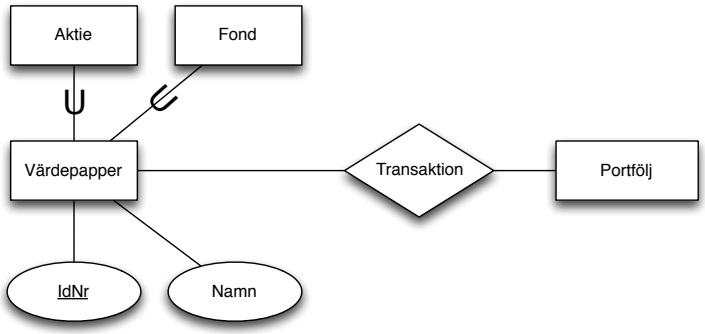
Attribut och samband som bara gäller för en subklass ritas dock ut på just den subklassen, så som i figur 1.15. För Chaufförerna är det viktigt att hålla reda på vilken körkortsbehörighet de har, och för konsulterna vilket timpris de har, samt vilka projekt de jobbar i.



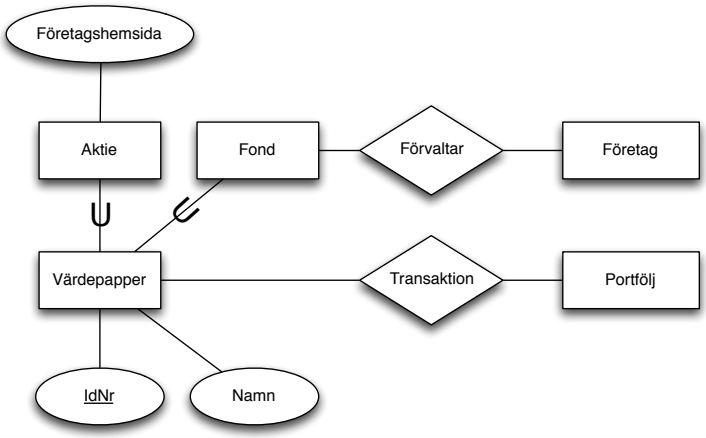
Figur 1.15: Samband och Attribut som bara gäller subklasserna (Chafför och Konsult) ritas vid respektive subklass.

Ett annat exempel finner vi i figur 1.16. Alla Värdepapper kan ingå i sambandet Transaktion, samt har namn och Id-nummer i databasen. Fonder ingår dessutom i ett samband Förvaltar med den fondförvaltare som sköter fonden, och för Aktierna finns en länk till företagets hemsida med i databasen, se figur 1.17.

I nästa avsnitt skall vi se hur man kan gå från ER-modell till Databasstruktur.



Figur 1.16: Aktie och Fond är subklasser till superklassen Värdepapper.



Figur 1.17: Samband och Attribut som bara gäller subklasserna (Aktie och Fond) ritas vid respektive subklass.

5 Från ER-modell till Databasstruktur

Hela poängen med ER-modeller är att de skall underlätta vid skapandet av databaser. I detta avsnitt kommer vi därför att beskriva hur man gör när man överför en ER-modell till en databasstruktur. Först skall vi emellertid se vad vi menar med *databasstruktur*.

5.1 Databasstruktur

Med databasstruktur menar vi en beskrivning av vilka tabeller som finns i databasen, och vilka primärnycklar dessa har. Primärnycklarna är det som är unikt för varje rad i databasen. Det är oftast en enskild kolumn, men ibland en kombination av kolumner.

Tidigare i detta dokument har vi tittat på tabeller, t.ex. tabellen Anställd:

Anställd				
AnsNr	Namn	Adress	Telefon	Lön
1	Kalle	Vintervägen 1	070-111111	20 000
2	Anna	Sommargatan 2	070-222222	40 000
3	Kim	Sommargatan 7	070-343434	40 000
⋮	⋮	⋮	⋮	⋮

För att slippa hitta på exempeldata, som Kalle, Anna och Kim ovan, så är det smidigt att ibland beskriva tabellen på följande vis: *Anställd(AnsNr, Namn, Adress, Telefon, Lön)*.

Vi kallar detta för *Databasstrukturen* för tabellen ovan. Den består helt enkelt av tabellens namn, samt kolumnnamnen, med primärnyckeln understruken (kom ihåg att flera kolumner ibland tillsamman utgör primärnyckeln, alla dessa är då understrukna, se t.ex. avsnitt 4.5.

Om vi istället har följande tabell

Dator		
InvNr	Typ	Tillverkningsår
1	Macbook	2013
2	Dell PC	2009
⋮	⋮	⋮

så skriver vi databasstrukturen: *Dator(InvNr, Typ, Tillverkningsår)*.

5.2 Sammanfattning av Reglerna

I nedanstående lista sammanfattas de 11 reglerna för översättning mellan ER-modell och Databasstruktur.

1. Varje vanlig entitetstyp blir tabell
2. 1:N-samband blir kolumn i N-tabellen
3. 1:1-samband blir kolumn i någon av tabellerna
4. N:M-samband blir egen tabell
5. Flervägssamband blir egen tabell
6. Attribut på samband blir kolumner
7. Svag entitetstyp blir tabell
8. Sammansatta attribut blir som delarna
9. Flervärt attribut blir egen tabell
10. Härledda attribut blir inte en tabell, utan en vy
11. Varje subklass blir egen tabell

Innan vi går igenom dessa regler i detalj, noterar vi att man kan sortera dem med avseende på om de resulterar i en tabell, eller en kolumn. Resultatet blir då följande, vilket kanske kan vara lättare att komma ihåg.

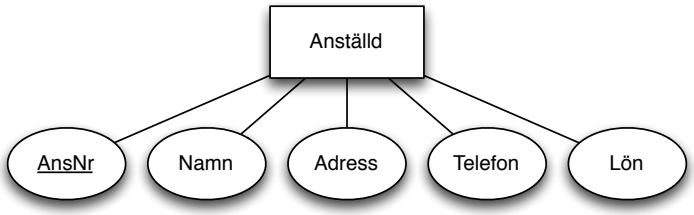
1. Egen tabell
   (a) Entitetstyper
   (b) Svaga entitetstyper
   (c) N:M samband
   (d) Flervägssamband
   (e) Flervärt attribut
   (f) Subklasser
2. Inte tabell
   (a) Härledda attribut (istället SQL-vy)
3. Kolumn i annan tabell
   (a) 1:N samband
   (b) 1:1samband
   (c) Attribut på samband

5.3 Reglerna i detalj

Regel 1: Varje vanlig entitetstyp blir tabell

I mer detalj lyder regeln:
Varje vanlig entitetstyp blir tabell, och attributen blir kolumner i tabellen.

Tillämpar vi denna regel på ER-modellen i figur 1.18 så får vi datastrukturen:
Anställd(AnsNr, Namn, Adress, Telefon, Lön).

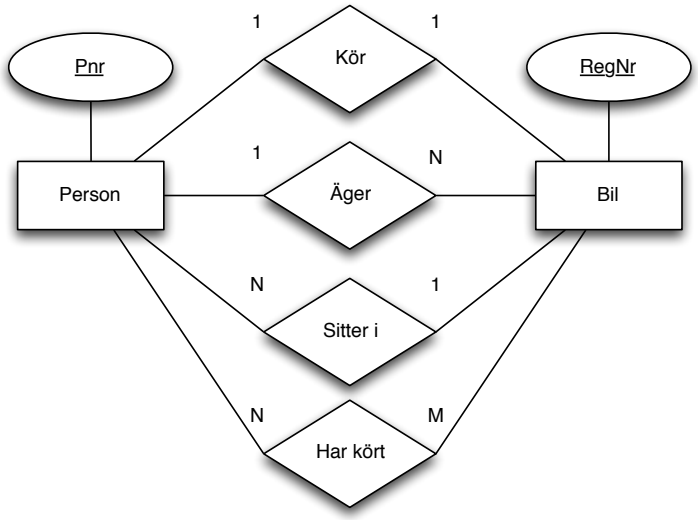


Figur 1.18: En ER-modell där regel 1 kan tillämpas.

Regel 2: 1:N-samband blir kolumn i N-tabellen

I mer detalj lyder regeln:
1:N-samband blir kolumn i tabellen på N-sidan av sambandet.

Tillämpar vi denna regel sambandet Äger i ER-modellen i figur 1.19 så får vi datastrukturen:
Person(Pnr)
Bil(RegNr, Ägare)



Figur 1.19: En ER-modell där regel 2, 3 och 4 kan tillämpas.

Notera här att kolumnen Ägare kommer att innehålla Pnr (Primärnyckeln) för personen som äger Bilen. För att göra detta tydligt kan vi i stället skriva databasstrukturen
Person(Pnr)
Bil(RegNr, Pnr)

Nu blir kopplingen tydlig mellan tabellerna. Men kopplingen inom tabellen blir istället mindre tydlig. Varför har bilen ett personnummer? När man ger namn till den nya tabellen får man alltså välja om man skall använda samma namn (Pnr) eller hitta på ett nytt (Ägare). I båda fallen kommer kolumnen att innehålla Pnr på ett antal ägare. Om vi som i figur 1.19 har flera samband måste vi ha olika namn på de kolumner som skapas av sambanden, eftersom kolumnerna i samma tabell inte får ha samma namn. På tentan är det viktiga att det framgår att ni vet vad ni gör.

Regel 3: 1:1-samband blir kolumn i någon av tabellerna

I mer detalj lyder regeln:

1:1-samband blir kolumn i någon av tabellerna, välj gärna den som ger minst Null-värden

.

Tillämpar vi denna regel sambandet Kör i ER-modellen i figur 1.19 så får vi databasstrukturen:
Person(Pnr)
Bil(RegNr, Körs av)

Oavsett vilken sida man väljer kommer det att bli många Null-värden, det är ju en massa människor som inte kör bilar just nu, och en massa bilar som inte körs. Men gissningsvis är det färre av det senare.

Regel 4: N:M-samband blir egen tabell

I mer detalj lyder regeln:

N:M-samband blir egen tabell. Primärnyckel i den nya tabellen blir kombinationen av primärnycklarna/nyckelattributen för de båda entiteterna

Tillämpar vi denna regel sambandet Har kört i ER-modellen i figur 1.19 så får vi databasstrukturen:
Person(Pnr)
Bil(RegNr)
Har kört(Pnr,RegNr)

Tillämpar vi regel 1-4 på hela ER-modellen så får vi databasstrukturen:
Person(Pnr, Sitter i)
Bil(RegNr, Körs av, Ägare)
Har kört(Pnr,RegNr)

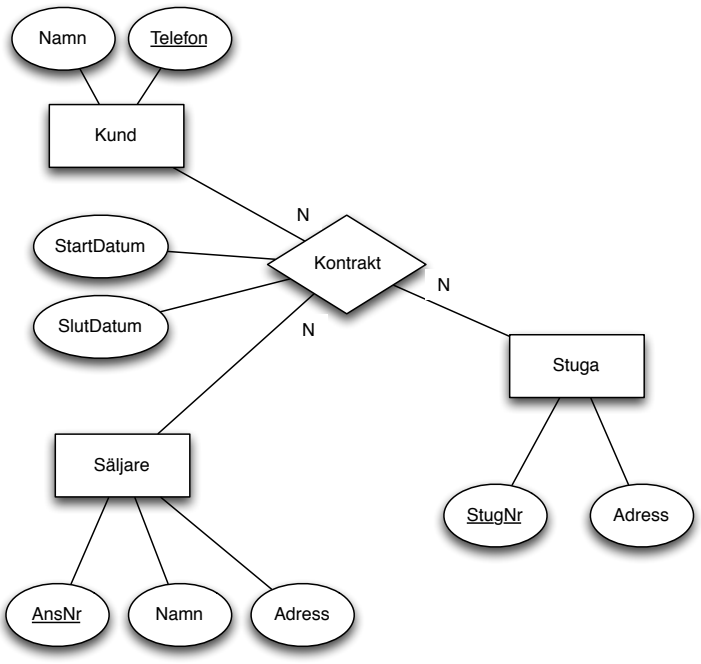
Regel 5: Flervägssamband blir egen tabell

I mer detalj lyder regeln:

Flervägssamband blir egen tabell. Primärnyckel i den nya tabellen blir kombinationen av primärnycklarna/nyckelattributen för de ingående entiteterna.

Tillämpar vi denna regel på sambandet Kontrakt ER-modellen i figur 1.20 så får vi databasstrukturen:
Kund(Telefon, Namn)
Stuga(StugNr, Adress)
Säljare(AnsNr, Namn, Adress)
Kontrakt(Telefon,StugNr,AnsNr, StartDatum, SlutDatum)

Notera att vi här även använt regel 6 nedan, för attributen StartDatum och SlutDatum.



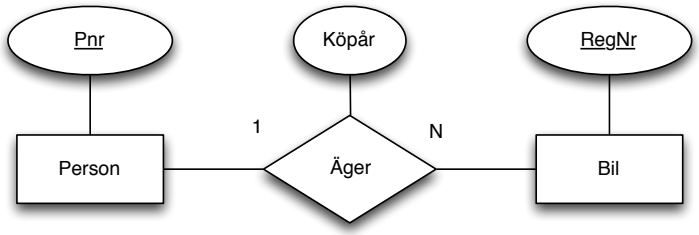
Figur 1.20: En ER-modell där regel 5 och 6 kan tillämpas.

Regel 6: Attribut på samband blir kolumner

I mer detalj lyder regeln:

Attribut på samband blir kolumner. I de fall då sambandet blir en egen tabell (regel 4 och 5) så hamnar attribut-kolumnerna i denna tabell. När sambandet i sig bara blivit en extra kolumn i en annan tabell (regel 2 och 3), så hamnar attribut-kolumnerna också i den tabellen.

I stycket ovanför så tillämpade vi denna regel på sambandet Kontrakt ER-modellen i figur 1.20. Tillämpar vi regeln på ER-modellen i figur 1.21 så får vi databasstrukturen:
Person(Prn)
Bil(RegNr, Ägare, Köpår)



Figur 1.21: En ER-modell där regel 6 kan tillämpas.

Regel 7: Svag entitetstyp blir tabell

I mer detalj lyder regeln:

Svag entitetstyp blir tabell. Primärnyckeln i tabellen utgörs av kombinationen av den svaga entitetens partiella nyckel och den identifierande entitettypens primärnyckel.

Vi påminner oss från avsnitt 4.5 att svaga entiteter är beroende av andra entiteter. För att slippa numrera alla kök i hela nyproduktionen hos NCC kan man tänka sig att de låter husen var identifierande entiteter, med fastighetsbetäckning som primärnyckel, och rummen i respektive hus vara svaga entiteter. Dessa identifieras genom en partiell nyckel (t.ex. kök, sovrum1, sovrum2, klädkammare ...) och det hus de tillhör.

Om vi tillämpar regel 7 på entiteten Rum i figur 1.22 så får vi databasstrukturen:
Hus(Fastighetsbetäckning)
Rum(Fastighetsbetäckning,Namn, Yta)

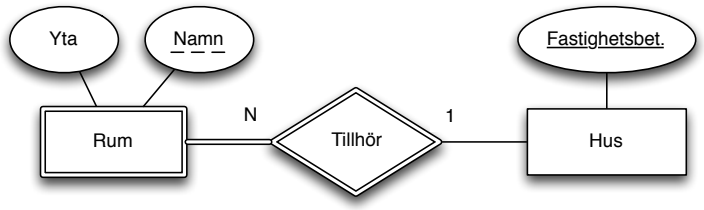
Notera att om entiteten inte varit svag, hade Fastighetsbetäckning ändå dykt upp i Rum-tabellen, som ett resultat av 1:N-sambandet (regel 2). Då hade det dock inte varit en del av primärnyckeln, utan bara en vanlig kolumn.

Regel 8: Sammansatta attribut blir som delarna

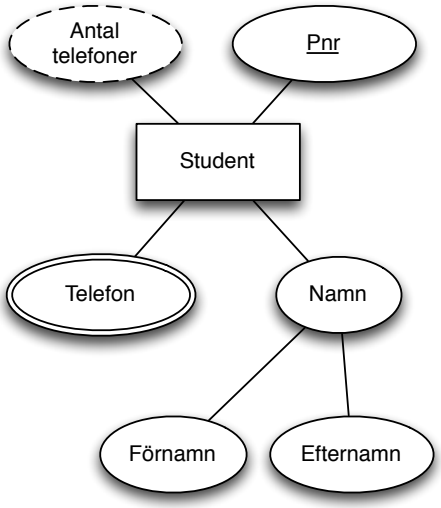
I mer detalj lyder regeln:

Sammansatta attribut blir som om del-attributen satt direkt på entiteten, och det sammansatta attributet inte hade existerat.

Tillämpar vi regeln på attributet Namn i ER-modellen från figur 1.23 så får vi databasstrukturen:
Student(Pnr, Förnamn, Efternamn).



Figur 1.22: En ER-modell där regel 7 kan tillämpas.



Figur 1.23: En ER-modell där regel 8, 9 och 10 kan tillämpas.

Regel 9: Flervärt attribut blir egen tabell

I mer detalj lyder regeln:

Flervärt attribut blir egen tabell. Primärnyckeln för tabellen är kombinationen av entitetens primärnyckel och det flervärda attributet.

Tillämpar vi regeln på attributet Telefon i ER-modellen från figur 1.23 så får vi databasstrukturen:
Student(Pnr)
Telefon(Pnr,TelefonNr).

Notera att detta gör att samma person kan ha hur många telefoner som helst, men även att samma telefon kan förekomma hos flera personer (om de t.ex. bor i samma lägenhet).

Regel 10: Härledda attribut blir inte en tabell, utan en vy

I mer detalj lyder regeln precis som ovan:

Härledda attribut blir inte en tabell, utan en vy.

Vyer är en mycket viktig del av databasen. De innehåller dock inga egna data, och är därför inte en del av

Databasstrukturen.

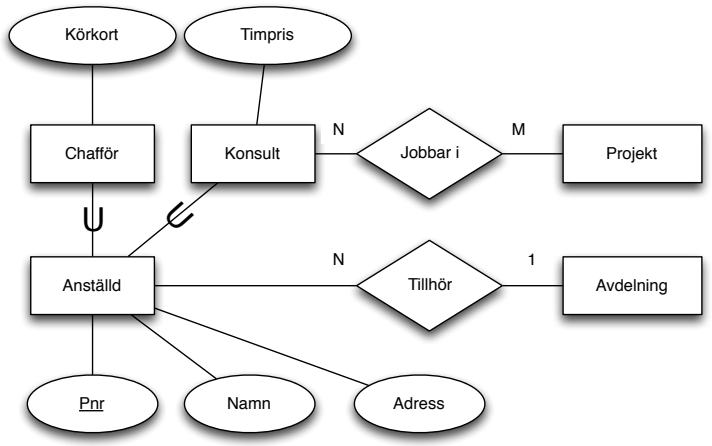
Om vi skulle överföra hela ER-modellen i figur 1.23 till en databasstruktur får vi sammanfattningsvis databasstrukturen:
Student(Pnr, Förnamn, Efternamn)
Telefon(Pnr,TelefonNr).

Regel 11: Varje subklass blir egen tabell

I mer detalj lyder regeln:

Varje subklass blir egen tabell. Primärnyckel är samma som superklassens primärnyckel. Övriga tabeller ges av de attribut och samband som gäller just den subklassen.

Om vi överför ER-modellen i figur 1.24 till en databasstruktur får vi databasstrukturen:
Anställd(Pnr, Namn, Adress, Avdelning)
Chaufför(Pnr, Körkort)
Konsult(Pnr, Timpris)
Jobbar i(Pnr,ProjNr)



Figur 1.24: En ER-modell där regel 11 kan tillämpas.

Detta avslutar vår genomgång av reglerna, samt hela dokumentet.