Lösningsförslag - Modellering Databassystem

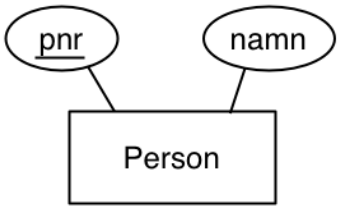
Christian Lennerholt

1 Modellering

Grundläggande förståelse:

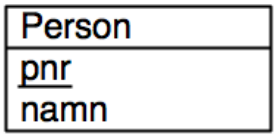
1.

ER-modell:



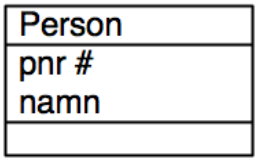
Använder en stark entitet för person eftersom endast personnumret identifierar en person. Personnummer används som primärnyckel och markeras med understrykning. Ett attribut för namn används då namn är en egenskap hos en person.

IE-modell:



En stark entitet symboliseras enligt ovan i EI-notationen. Primärnyckeln stryks under.

UML-modell:



Person representeras som en klass (klass är synonymt med entitet i de övriga notationerna) i UML enligt ovan. I den övre rutan erhålls klassens namn. I mitten erhålls alla attribut och nyckeln symboliseras med #.

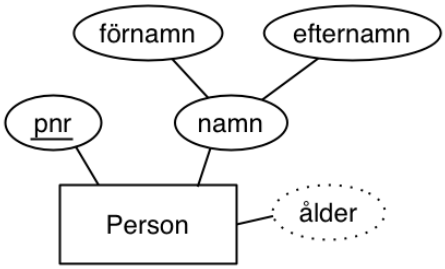
Relationsdatamodell:

Person (pnr , namn)

Entitetens (ER) / klassens (UML) namn står skrivet innan parentes. Eftersom personnummer är primärnyckel stryks den under. Namn är ett attribut.

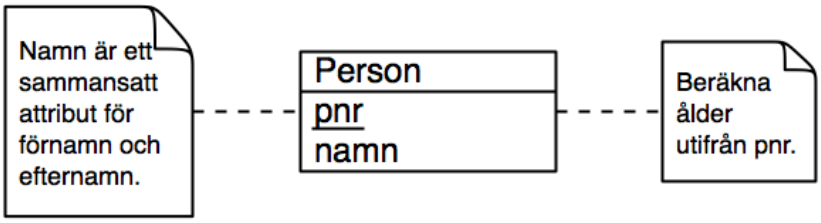
2.

ER-modell:



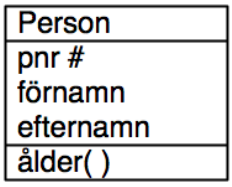
För- och efternamn bildar ett sammansatt attribut. Ålder är ett härlett attribut då det går att räkna ut personens ålder med hjälp av personnumret.

IE-modell:



Noteringar används för att visa sammansatta och härledda attribut.

UML-modell:



För och efternamn läggs till som vanliga attribut. Att kunna beräkna ålder representeras som en funktion och syns i den nedre rutan i UML-klassen. Parentes symboliserar funktionen.

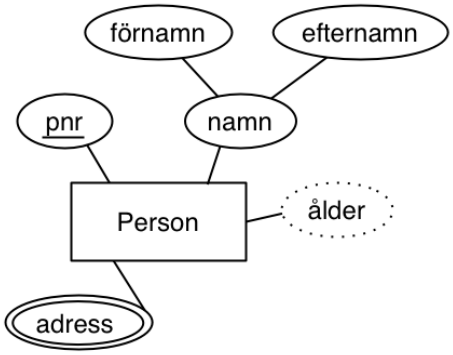
Relationsdatamodell:

Person (pnr , förnamn , efternamn)

Ett sammansatt attribut lagrar endast specifika data, i detta fall förnamn och efternamn. Ålder som kan beräknas lagras inte i databasen utan visar att ålder ska kunna beräknas i den applikation databasen ska användas till, exempelvis en hemsida eller program.

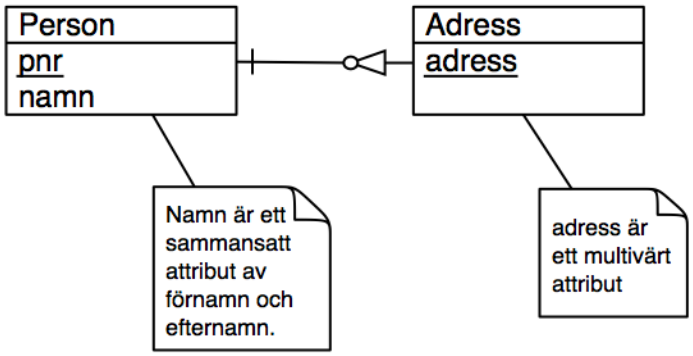
3.

ER-modell:



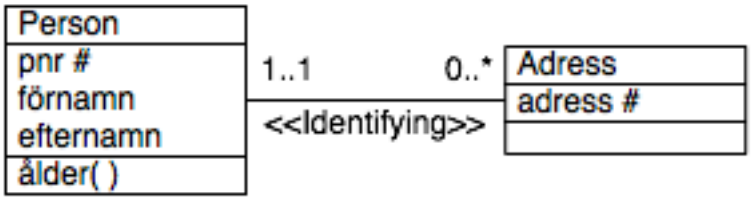
Eftersom en person kan ha flera bostäder används ett multivärt attribut för att representera alla adresser. Dubbelstreck visar ett multivärt attribut. Observera att under föreläsningen instruerades att en egen tabell (kardinalitet N till M) skapas för att representera ett multivärt attribut. Båda alternativen är korrekta.

IE-modell:



Ett multivärt attribut representeras med en svag entitet som relaterar till person. Symbolen  används för att representera en svag relation, dvs. en identifierande relation. Det måste finnas en person för att det ska kunna existera en adress. Skapar också en notering som säger att adress är ett multivärt attribut.

UML-modell:



Ett multivärt attribut representeras som en svag relation. En klass för adress skapas. En svag entitet/klass representeras med en identifierande relation med symbolen <<Identifying>> som skrivs under relationsstrecket. Detta innebär att adress identifieras av adress och personens personnummer (från personklassen).

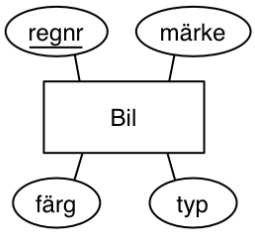
Relationsdatamodellen:

Person (pnr , förnamn , efternamn)
Adress (pnr , adress)

Adress lagras i en egen relation där pnr och adress tillsammans utgör nyckel. Eftersom pnr refereras till persontabellen blir den även främmande nyckel i adressrelationen, vilket syns då pnr har ett streck över sig.

4.

ER-modell:



Entiteten består av en nyckel och dess attribut.

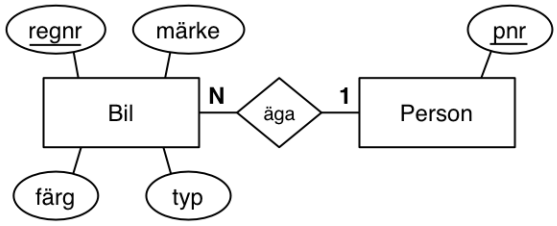
IE-modell:

UML-modell:

Relationsdatamodell:
Bil (regnr , märke , färg , typ)

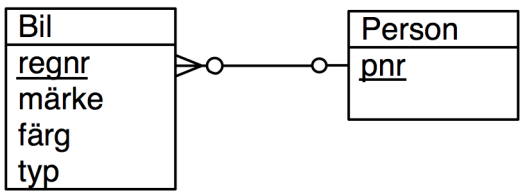
5.

ER-modell:



En till entitet för person skapas. Eftersom det saknas information om person i uppgiftsbeskrivningen så måste vi alltid minst anta vad som identifierar entiteten. Vi får aldrig skapa entiteter utan en nyckel! I detta fall identifieras en person av dess personnummer. Eftersom det inte framgår några egenskaper hos personen används inga attribut. Relationen ”äga” används för att visa att en person kan äga många bilar samt att en bil endast kan ägas av en person. ”Kardinaliteten” blir därför av typen ”många till en”. Det är också tillåtet att skriva ut vad relationen syftar till. I detta fall står det ”äga” för att förtydliga att en bil ägs av en person. Det är också tillåtet att lämna relationen tom, dvs. att inte skriva ”äga”.

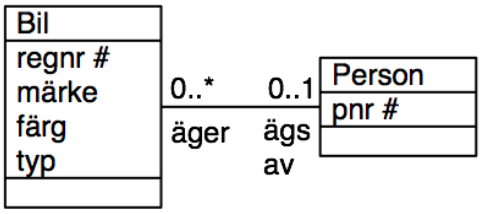
IE-modell:



Kardinaliteten av relationen utgörs av symbolerna ovan. På vänstra sidan visar symbolen ”0 eller många” och höger sida ”0 eller 1”. Detta innebär

att en bil kan ägas av 0 eller 1 person. En person kan äga 0 eller många bilar.

UML-modell:



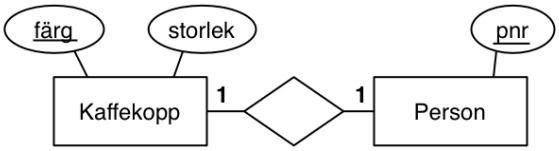
Relationen består endast av ett streck och kardinaliteten står ovan, dvs. en bil kan ägas av 0 eller 1 person. En person kan äga 0 eller flera bilar. Vad relationen syftar till kan tydliggöras under strecket.

Relationsdatamodell:

Person (pnr)
Bil (regnr , märke , färg , typ , pnr)

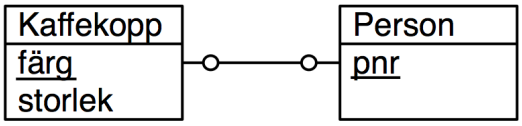
Båda entiteterna får sina egna relationer. Vid en relation med kardinaliteten ”många till en” ska den ”största” sidan erhålla den ”mindre” sidan som främmande nyckel. Detta innebär alltså att nyckeln för entiteten person ska vara en främmande nyckel i entiteten bil, vilket syns då pnr har ett streck ovan sig. Observera att pnr i bilentiteten inte är understruken eftersom pnr inte är en del av primärnyckeln. Det räcker alltså med att veta regnr på en bil för att kunna identifiera en bil. Därför är regnr ensam primärnyckel för entiteten bil.

6. ER-modell:



Eftersom endast en person kan äga en kaffekopp samt att en kopp endast ägs av en person används kardinaliteten ”en till en”.

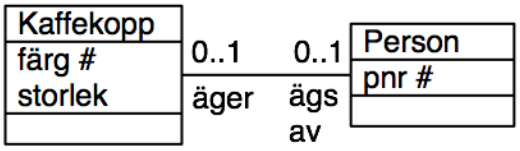
IE-modell:



Ringarna på relationen symboliserar en ”en till en”-relation som inte är tvingande, dvs. en person måste inte äga en kaffekopp och tvärtom.

Istället kan en person äga en kaffekopp samt en kaffekopp kan ha en ägare.

UML-modell:



Kardinaliteten blir 0..1.

Relationsdatamodell:

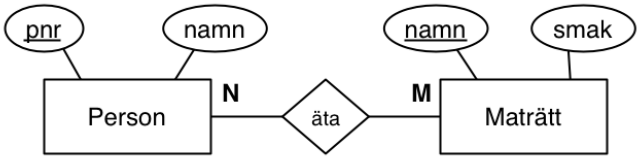
Person (pnr)
Kaffekopp (färg , storlek , pnr)

Eller

Kaffekopp (färg , storlek)
Person (pnr , färg)

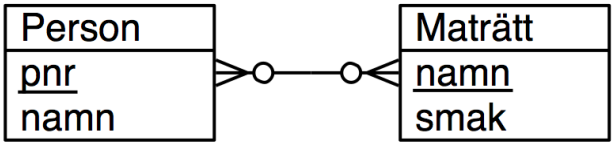
I de fall kardinalitet av typen ”en till en” förekommer så kan vi välja vilken sida som kommer bestå av en främmande nyckel. Antingen erhåller relationen kaffekopp en främmande nyckel från person eller så erhåller person en främmande nyckel från kaffekopp. Tänk på att en främmande nyckel refererar till primärnyckeln av en entitet.

7. ER-modell:



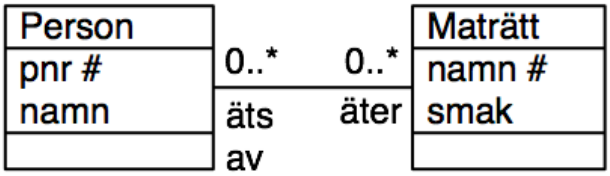
Eftersom en person kan äta många olika maträtter samt att en maträtt kan ätas av många olika personer används en relation med kardinaliteten ”många till många”, vilket skrivs ut som ”N” till ”M”.

IE-modell:



En relation av kardinalitet ”många till många” representeras av två ”kråkfötter” enligt ovan.

UML-modell:



Kardinaliteten ”många till många” symboliseras av 0..* i UML.

Relationsdatamodell:

Person (pnr , namn)
Maträtt (namn , smak)
Äta (pnr , namn)

En relation med kardinaliteten av typen ”många till många” skapas en egen relation där de tillhörande entiteternas nycklar hämtas. I detta fall hämtas pnr från person och namn från maträtt och blir primärnycklar i äta-relationen. Eftersom de kommer från andra entiteter blir de också främmande nycklar i äta-relationen. Observera att det är två stycken streck som symboliserar främmande nycklar, vilket visar att nycklarna kommer från två olika entiteter. Det är alltså inte tillåtet att använda ett gemensamt streck enligt bilden nedan för att representera främmande nycklen(för då tror vi att pnr och namn tillsammans är primärnyckel från en entitet, vilket är helt felaktigt i detta exempel). Däremot är det tillåtet att ha ett gemensamt streck som symboliserar primärnycklarna, se nedan.

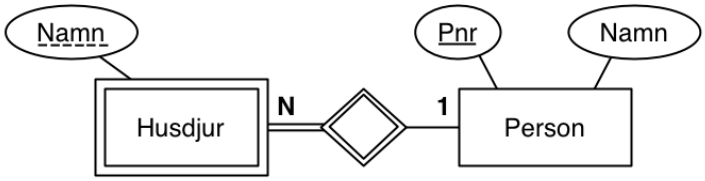
Vanligt misstag:

Person (pnr , namn)
Maträtt (namn , smak)
Äta (pnr , namn)

Gemensam understrykning för primärnyckel (tillåtet):

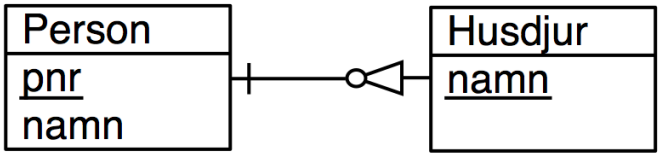
Person (pnr , namn)
Maträtt (namn , smak)
Äta (pnr , namn)

8. ER-modell:



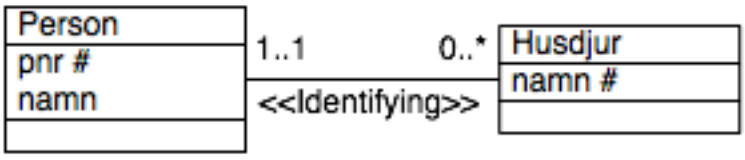
Eftersom husdjur identifieras med dess namn och personen som äger husdjuret bildas husdjur som en svag entitet, dvs. husdjurets namn räcker inte för att identifiera husdjuret. Först i kombination med personen som äger husdjuret kan husdjuret identifieras. Exempelvis kan flera ägare ha en hund med samma namn. Syntaxen för en svag entitet är dubbla streck runt entiteten. Husdjurets namn som utgör partiell nyckeln streckas. Hela nyckeln är namn och personnummer tillsammans, se relationsdatamodellen. En svag entitet måste alltid ha minst en identifierande relation, dvs. en relation till den/de entiteter som hjälper till att identifiera husdjuret (I detta fall är det personentiteten). En identifierande relation symboliseras med en dubbelstreckad relations-romb. Eftersom en person måste finnas för att kunna identifiera ett husdjur används dubbelstrecket mellan husdjur och relationsromben. Detta dubbelstreck symboliserar en tvingande relation. Kardinaliteten vid en identifierande relation kommer alltid att vara av typen ”många till en”.

IE-modell:



En svag entitet representeras av ovanstående relationssymbol. ”Det liggande spöket” innebär att husdjur är svag mot person. Ettan på vänstra sidan visar att det är en tvingande relations. Alltså, det måste finnas en person för att det ska kunna existera ett husdjur.

UML-modell:



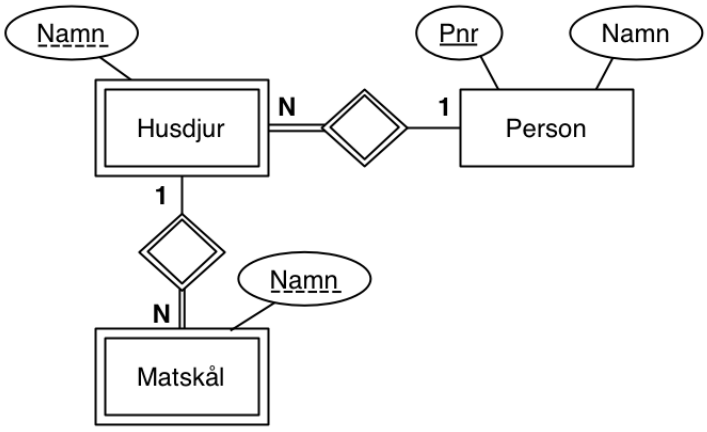
Eftersom husdjur är en svag relation identifieras den av sitt namn och personen som äger husdjuret. Relationensom kopplas ihop med personklassen symboliserar en svag relation eftersom <<Identifying>> skrivs under strecket. Husdjurets namn och personens personnummer utgör nyckel för ett husdjur.ns nyckel. Kardinaliteten kommer alltid att vara 1..1 och 0..*.

Relationsdatamodell:

Person (pnr , namn)
Husdjur (namn , pnr)

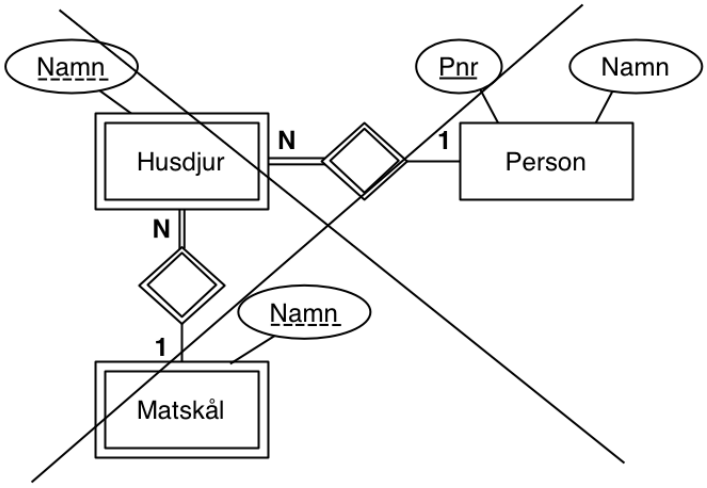
En tumregel är att alltid börja med att skapa relationstabeller för de starka entiteterna följt av de svaga entiteterna. I detta fall börjar vi att skapa person som har pnr som primärnyckel och namn som attribut. Eftersom husdjur är en svag entitet vet vi att primärnyckel måste bestå av minst två nycklar. I detta fall namnet på husdjur samt personentitetens primärnyckel pnr. Eftersom pnr kommer från en annan entitet markeras den också som främmande nyckel genom ett streck ovanför.

9. ER-modell:



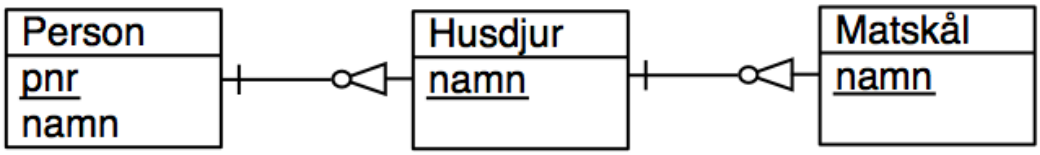
Matskål bildas som en svag entitet eftersom den identifieras av dess namn och i kombination med husdjuret som äter ur skålen. Den identifierande relationen skapas mellan matskål och husdjur. Observera att den tvingande relationen går från matskål till husdjur, vilket innebär att för att kunna identifiera en matskål måste ett husdjur finnas. Ett vanligt misstag är att sätta den tvingande relationen på fel sida, se nedan.

Vanligt misstag:



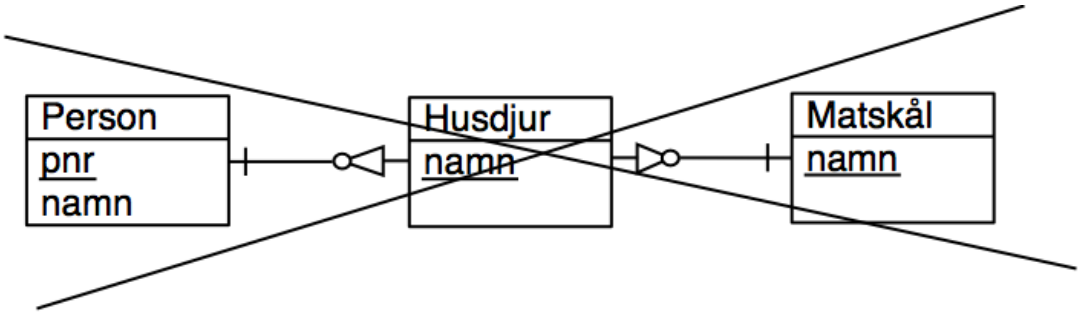
Ovanstående lösning är felaktig då husdjur har två identifierande relationer. Det innebär att för att kunna identifiera ett husdjur måste vi också veta vem som äger husdjuret samt vilken matskål som husdjuret äter från. Om vi då inte vet matskålen kan vi alltså inte identifiera ett husdjur, vilket låter helt orimligt. Ett annat fel som förekommer är att matskål inte har någon identifierande relation, vilket är ett krav att det måste finnas för en svag entitet.

IE-modell:



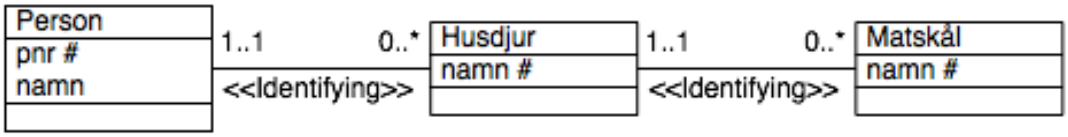
Viktigt att relationerna hamnar åt rätt håll. Enligt ovan identifieras matskål av husdjuret och dess ägare.

Vanligt misstag:



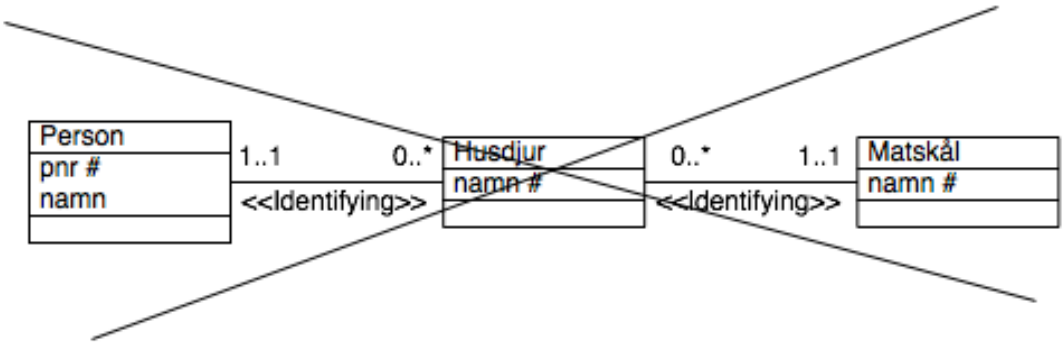
Ett vanligt misstag är att vända på relationsymbolerna. Detta får en stor felaktig konsekvens. Enligt ovan skulle ett husdjur identifieras av sitt namn och personens personnummer samt matskålens namn, vilket är helt felaktigt. Därför är det viktigt att vara noggrann när relationerna sätts ut.

UML-modell:



Samma princip som tidigare, dvs. relationen kopplas till den klass som används för att identifiera den svaga klassen. Husdjur identifieras av sitt namn och personens personnummer. Matskålen identifieras av sitt namn tillsammans med husdjurets namn och dess ägare. Viktigt att kardinaliteten blir rätt. Det måste finnas ett husdjur för att en matskål ska få finnas.

Vanligt misstag:



Ett vanligt misstag är att sätta kardinaliteten fel för en identifierande relation. I ovanstående lösnining skulle ett husdjur identifieras av sitt namn i kombination med personens personnummer och matskålens namn, vilket är felaktigt enligt uppgiftsbeskrivningen.

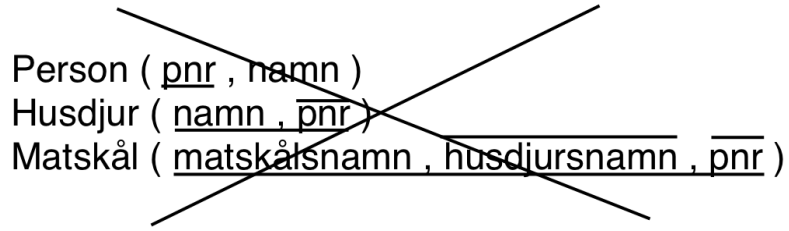
Relationsdatamodell:

Person (pnr , namn)
Husdjur (namn , pnr)
Matskål (matskålsnamn , husdjursnamn , pnr)

En svårighet i detta exempel är att matskål identifieras av en svag entitet, vilket i sin tur identifieras av personen som äger husdjuret. Alltid när vi hämtar nycklar från en entitet så hämtas HELA dess primärnyckel. I detta fall är hela nyckeln i husdjursentiteten husdjurets namn och personens personnummer. Eftersom båda tillsammans är primärnyckel och kommer från en annan entitet (än matskål) markeras den som en främmande

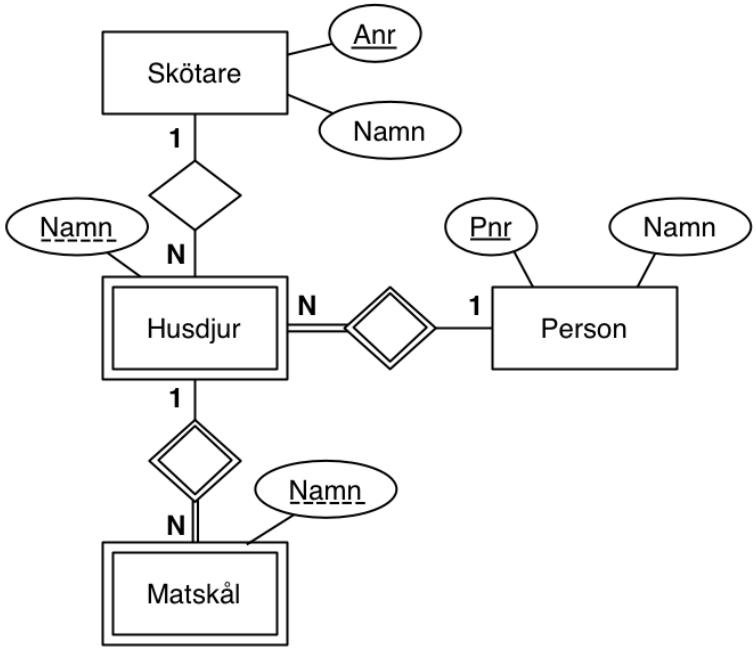
nyckel. Observera också att nycklarna har fått ett mer förtydligande namn i syfte att undvika att två nycklar får exakt samma namn. Husdjursnamn i entiteten matskål visar att namnet kommer från husdjur. Om vi inte förtydligat namnet hade ”namn” lagrats två gånger i matskålsentiteten, en som representerar matskålens namn och en som representerar husdjurets namn. Det är alltså tillåtet att förtydliga detta genom att ge ett mer symboliserande namn. Observera också att främmande nyckeln markeras som ett heldraget sträck eftersom både husdjurets namn och personnummer är tillsammans primärnyckel i husdjursentiteten. Ett vanligt misstag i detta läge är att markera den främmande nyckeln med två streck ovanför, se nedan.

Vanligt misstag:



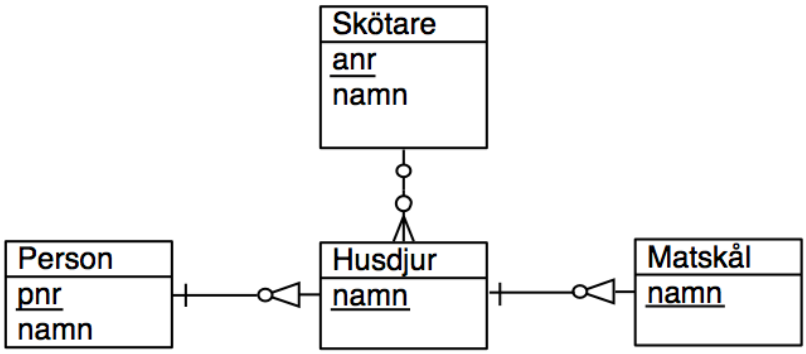
Om vi markerar främmande nyckeln med två streck ovanför husdjursnamn och personnummer så symboliseras det som att det är två stycken nycklar som refereras till två olika entiteter, vilket är felaktigt eftersom primärnyckeln i entiteten husdjur är husdjurets namn och personens personnummer. Tänk på att alltid referera till hela nyckeln och där hela nyckeln symboliseras med ett heldraget streck ovanför.

10. ER-modell:

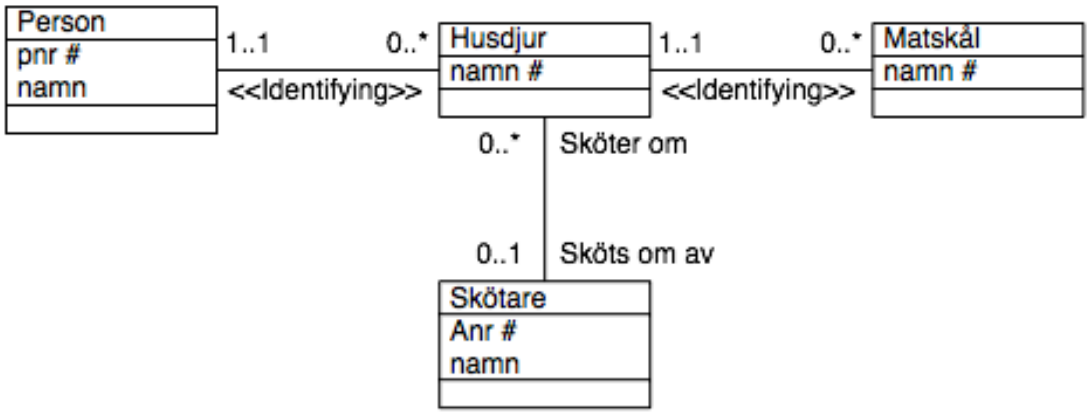


Skötare är en stark entitet eftersom det räcker med att veta anställningsnumret för att kunna identifiera skötare. Då en skötare kan ta hand om flera husdjur och ett husdjur kan ha en skötare är kardinaliteten av typ ”många till en”. Observera att det är en vanlig relation som används eftersom. Ett vanligt misstag är att använda en identifierande relation även här bara för att husdjur är en svag entitet, vilket är felaktigt. Va noga med att skilja på en identifierande relation och en vanlig relation.

IE-modell:



Här skapas en egen entitet för skötare. Kardinalitetet blir ”många till en”.
UML-modell:



En klass som representerar skötare skapas och en relation till husdjur med kardinaliteten ”många till en” används för att lösa uppgiften.

Relationsdatamodell:

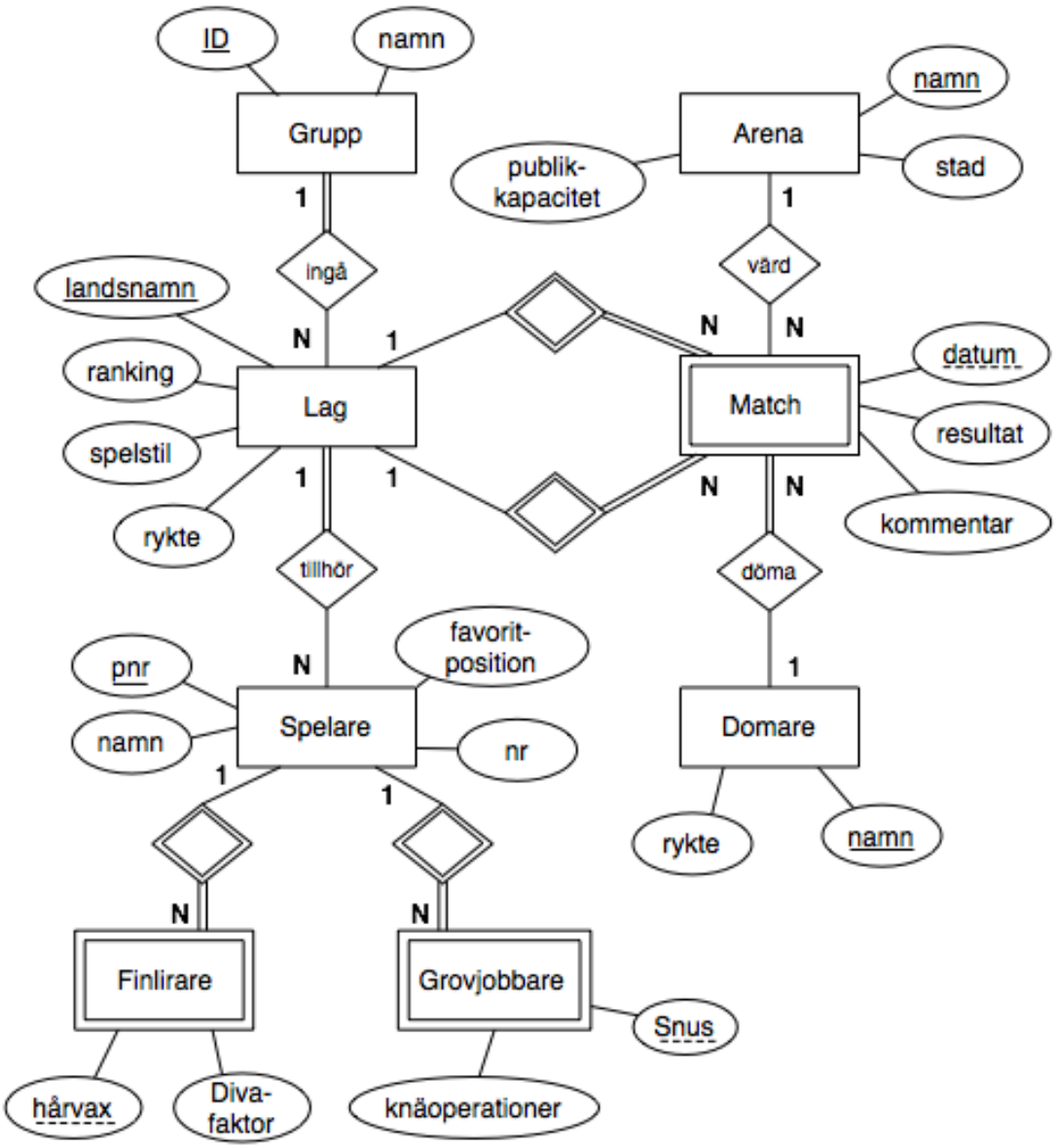
Person (pnr , namn)
Husdjur (namn , pnr , anr)
Matskål (matskålsnamn , husdjursnamn , pnr)

Tänk på att få främmande nycklar korrekt. I entiteten husdjur ser vi att den har två stycken främmande nycklar. Den ena är för den identifierande relationen. Den andra för att kardinaliteten är av typ ”många till en”, se

övning 5 ovan. Tumregeln är att vi då ska ha två stycken streck som symboliserar främmande nycklar i husdjurstabellen som vi skapar.

Fördjupning:

11. ER-modell:

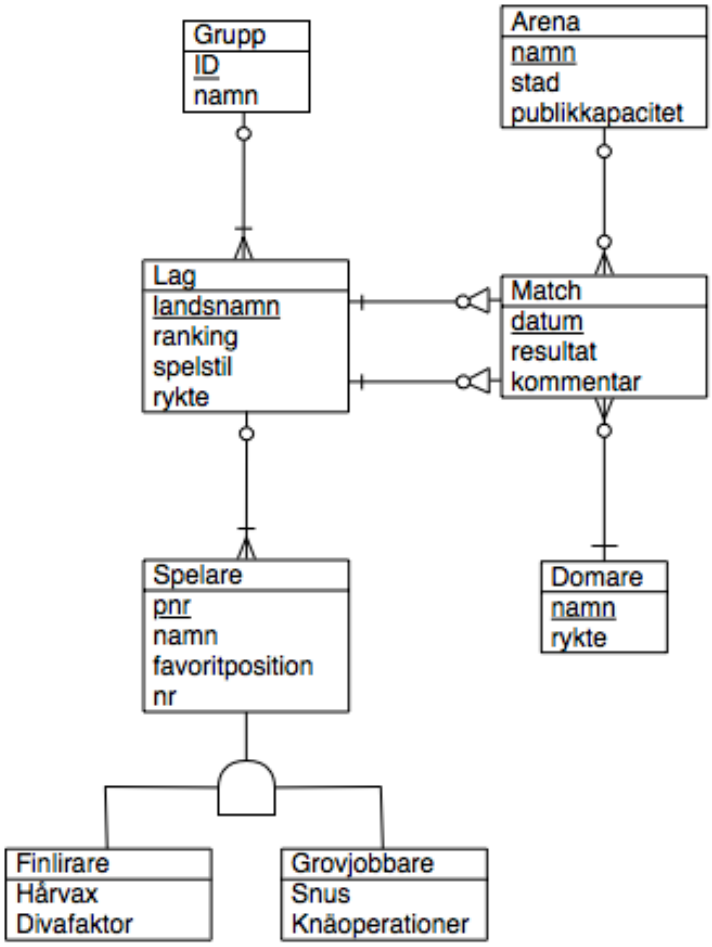


En av svårigheterna i övningen är hur en match alltid består av två lag. I detta fall används två identifierande relationer till entiteten lag, där varje relation motsvarar ett lag. I övrigt gäller det att förstå hur kardinaliteten blir för varje relation. I vissa fall används ”tvingande” relationer då det uttryckligen står i uppgiftsbeskrivningen att något måste finnas, exempelvis en grupp består alltid av ett antal lag, ett lag består alltid av ett antal spelare samt att en match alltid har en domare.

En spelare kan vara antingen finlirare eller grovjobbare (även samtidigt), vilket karakteriseras av ett arv. Detta medför problematik vid ER-modellering eftersom arv inte stöds i modelleringsnotationen. Det innebär att det inte går att lösa detta krav från uppgiftsbeskrivning. Istället för att lämna detta krav blankt får vi designa om. Ovan syns att två

svaga entiteter skapas för att representera finlirare och grovjobbare. Detta ska ses som en nödlösning som inte är helt optimal. Observera att detta är ett sätt att lösa arvet på bästa sätt med ER-notationen. Det går istället att skapa en 1-1 koppling eller lägga alla attribut direkt i entiteten spelare. Men detta anses fortfarande som en mindre bra lösning eftersom ER-notationen inte har ett bra stöd för arv. Modelleringsnotationerna IE och UML har stöd för arv och lämpar sig bättre för att lösa arv.

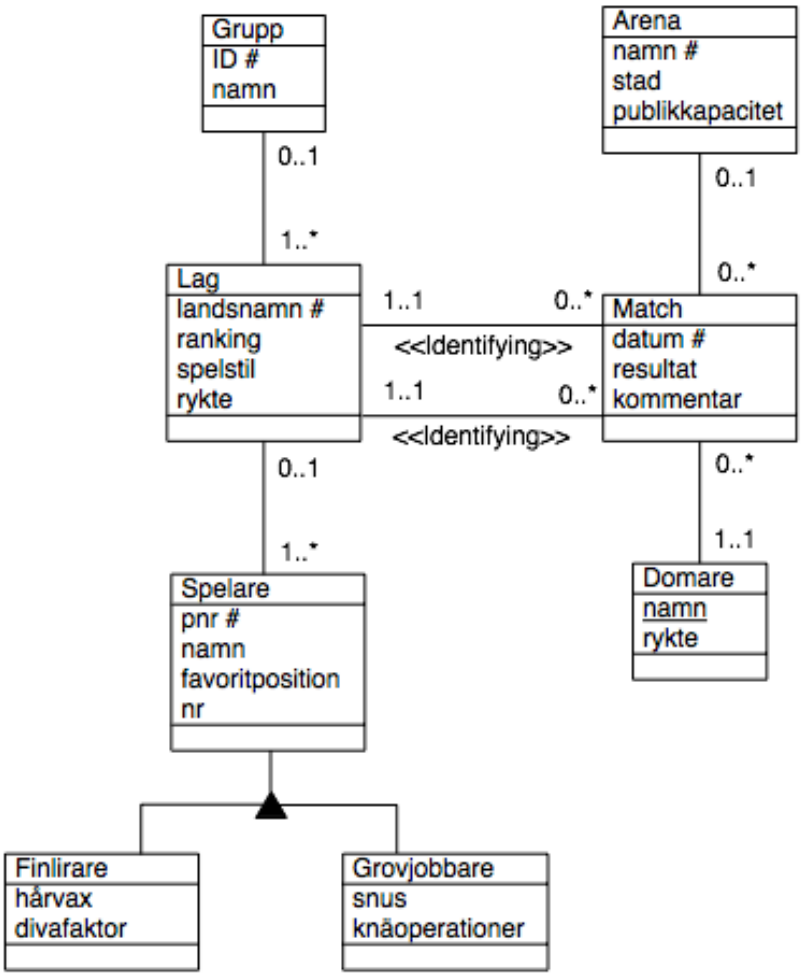
IE-modell:



Samma grundtänk som för ER-modellering, men här skiljer sig notationen när det gäller svaga relationer och tvingande relationer. Observera skillnaden när relationerna består av en 0 eller I. Ex, en grupp måste bestå av minst ett eller flera lag osv.

Arv stöds i IE-notationen, vilket underlättar när vi löser uppgiftsbeskrivningens krav om spelare som är finlirare eller grovjobbare (eller båda samtidigt).

UML-modell:



Samma grundprincip som de övriga modellnotationerna. Viktigt att du känner dig bekväm med hur de olika detaljerna representeras i de olika modellerna. Exempelvis, svaga relationer/entiteter, tvingande relationer, kardinalitet och arv. En förvirrande detalj gällande arv och skillanden mellan IE och UML notationerna är att IE en ofylld symbol för att representera överlapp, dvs. en spelare kan vara finlirare och grovjobbare samtidigt. I UML är symbolen ifylld för att representera överlapp.

Relationsdatamodell, endast för ER-modellen:

Grupp (id , namn)
Lag (landsnamn , ranking , spelstil , rykte , id)
Spelare (pnr , namn , favoritposition , nr , lag)
Arena (namn , stad , publikkapacitet)
Domare (namn , rykte)
Match (datum , lag1 , lag2 , resultat , kommentar , domare , arena)
Finlirare (hårvax , pnr , divafaktor)
Grovjobbare (snus , pnr , knäoperationer)

En rekommendation är att alltid börja med alla starka entiteter, sedan svaga entiteter och sist kontrollera att alla relationer blir korrekta med dess främmande nycklar. I de fall en entitet har en främmande nyckel som refererar till en entitet är det tillåtet att använda entitetens namn istället för namnet på dess nyckel. I detta fall tar vi för givet att entiteten har en nyckel. Exempel: för relationstabellen ”spelare” används en främmande nyckel ”lag”. Eftersom lag har en primärnyckel: landsnamn, är det denna vi refererar till. Ett alternativ är att istället för att använda ”lag” som främmande nyckel kan vi återge hela primärnyckelns namn: landsnamn, istället om vi vill:

Spelare (pnr , namn , favoritposition , nr , landsnamn)

Huvudsaken är att det är tydligt för vad datan representerar. Däremot måste vi alltid skriva ut alla nycklar om mer än en nyckel förekommer för en entitet (exvis svag entitet).

Var noga med att få med alla främmande nycklar på ett korrekt sätt. Observera skillnaderna mellan ett heldraget streck ovanför främmande nycklar. Ett streck per relation. En främmande nyckel blir också primärnyckel i de fall en identifierande relation används, samt vid relationer med kardinalitet av typ ”många till många”.

Relationsdatamodell, för IE och UML:

Grupp (id , namn)
Lag (landsnamn , ranking , spelstil , rykte , id)
Arena (namn , stad , publikkapacitet)
Domare (namn , rykte)
Match (datum , lag1 , lag2 , resultat , kommentar , domare , arena)

För att kunna representera spelare och dess arv finns 3 olika lösningar som lämpar sig (lägg till någon av dessa med ovanstående för en komplett relationsdatamodell):

1)

Spelare (pnr , namn , favoritposition , nr , lag)
Finlirare (pnr , hårvax , divafaktor)
Grovjobbare (pnr , snus , knäoperationer)

Här skapas en relation för varje entitet. Subtyperna är kopplade till supertypen mha. främmande nycklar.

2)

Finlirare (pnr , namn , favoritposition , nr , lag , hårvax , divafaktor)
Grovjobbare (pnr , namn , favoritposition , nr , lag , snus , knäoperationer)

Mindre entiteter, vilket kan snabba upp databasen beroende på vad den används till. Här läggs data från ”spelare”-entiteten in i subtyperna direkt.

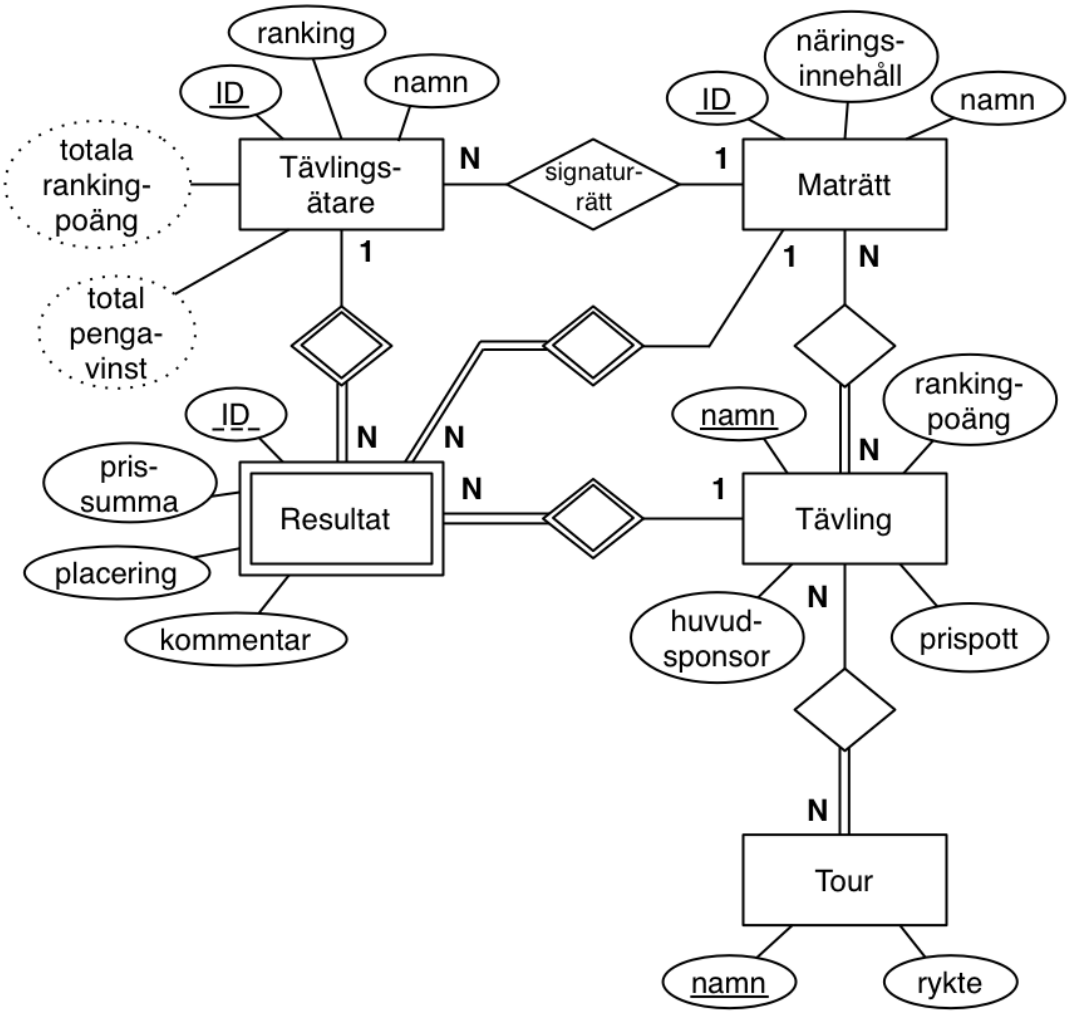
3)

Spelare (pnr , namn , favoritposition , nr , tag , spelartyp , hårvax , divafaktor, snus , knäoperationer)

Här läggs alla data i supertypen ”spelare”. Eftersom allt lagras som attribut innebär det att det kommer lagras väldigt mycket NULL-värden.

Observera att det fjärde sättet att representera arv som visas i föreläsningsmaterialet inte lämpar sig för uppgiftens arv. För det fjärde sättet används ett attribut för att avgöra om man är finlirare eller grovspelare. Eftersom det är tillåtet att vara både och samtidigt enligt uppgiftsbeskrivning fungerar inte det fjärde sättet att representera arvet.

12. ER-modell:



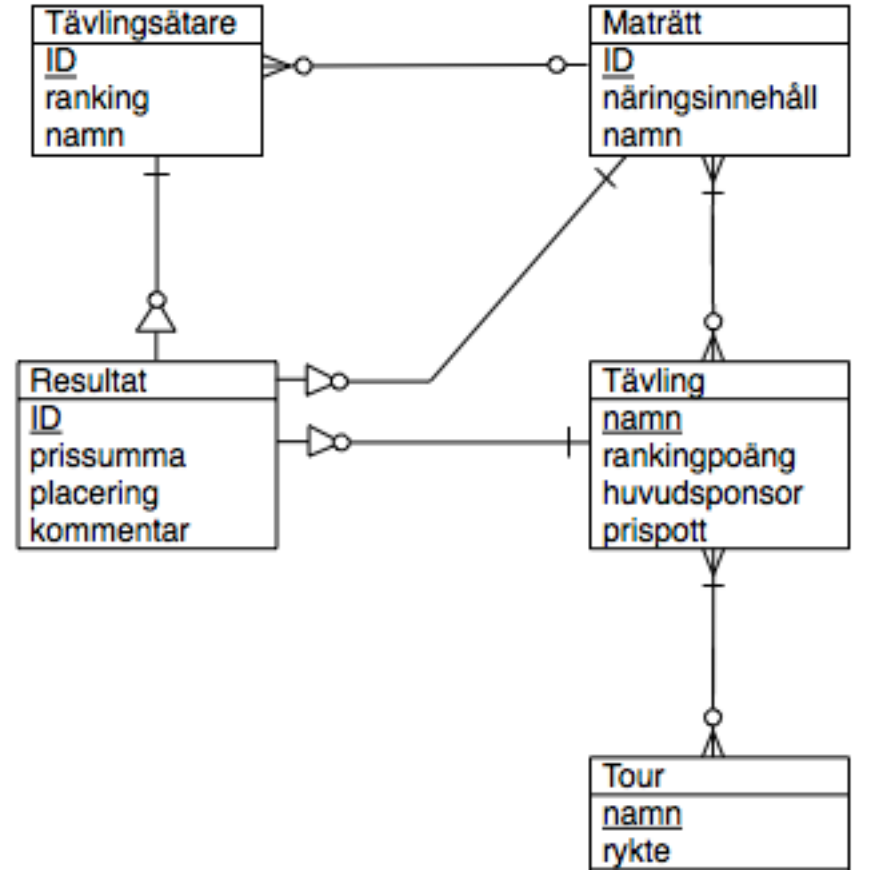
Totala rankingpoäng och pengavinst ska beräknas fram och symboliseras med en streckad linje. En tävlingsätares signaturrätt uppfylls genom en

relation till maträtt med kardinalitet ”många till en” Detta innebär att maträttens ID kommer lagras som en främmande nyckel i tävlingsätare och symboliserar därför signaturrätt.

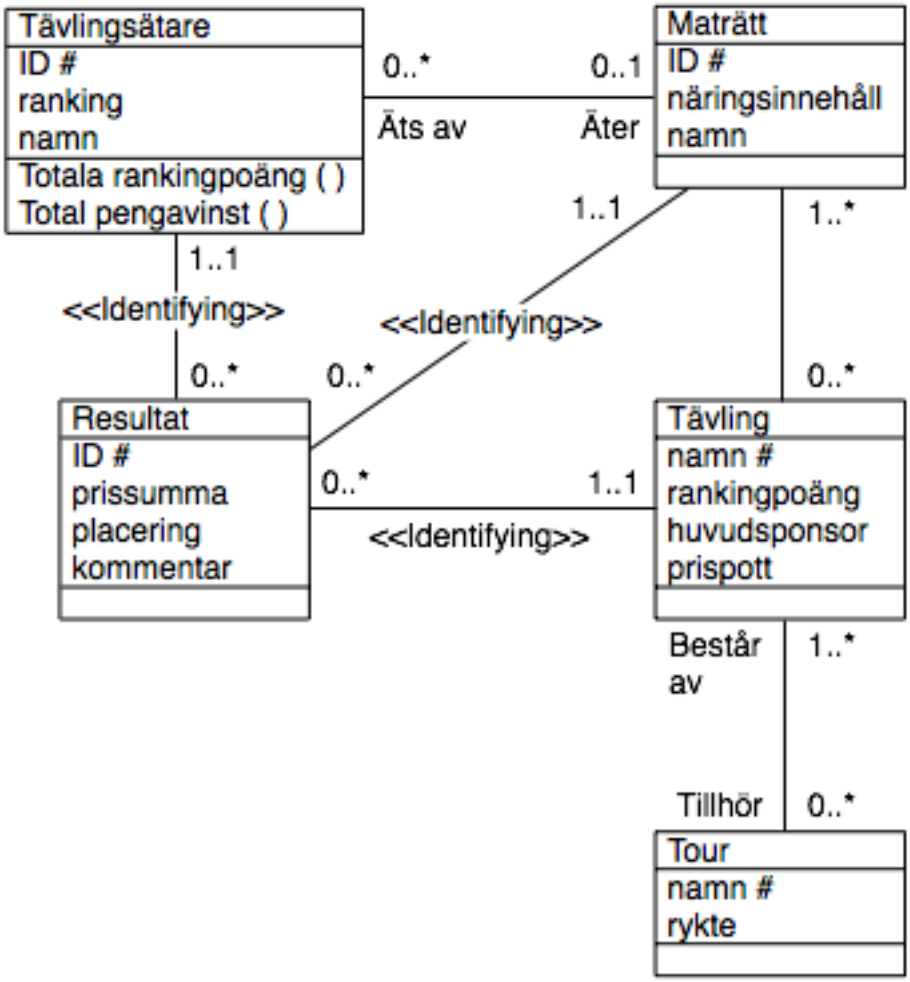
Resultatet är en svag entitet då den identifieras av dess ID-nummer i kombination med tävlingsätarens ID, maträttens ID och tävlingens namn. Detta syns eftersom 3st identifierande relationer används till de tillhörande entiteterna.

En tävling måste bestå av minst en maträtt och symboliseras med en tvingande relation (dubbelstreck). Relationen kan tyckas lite onödig då resultat innehåller både tävling och maträtt. Men för att vara tydlig (att en tävling minst består av en maträtt) skapas en relation mellan tävling och maträtt med kardinalitet ”många till många”.

IE-modell:



UML-modell:



Relationsdatamodell:

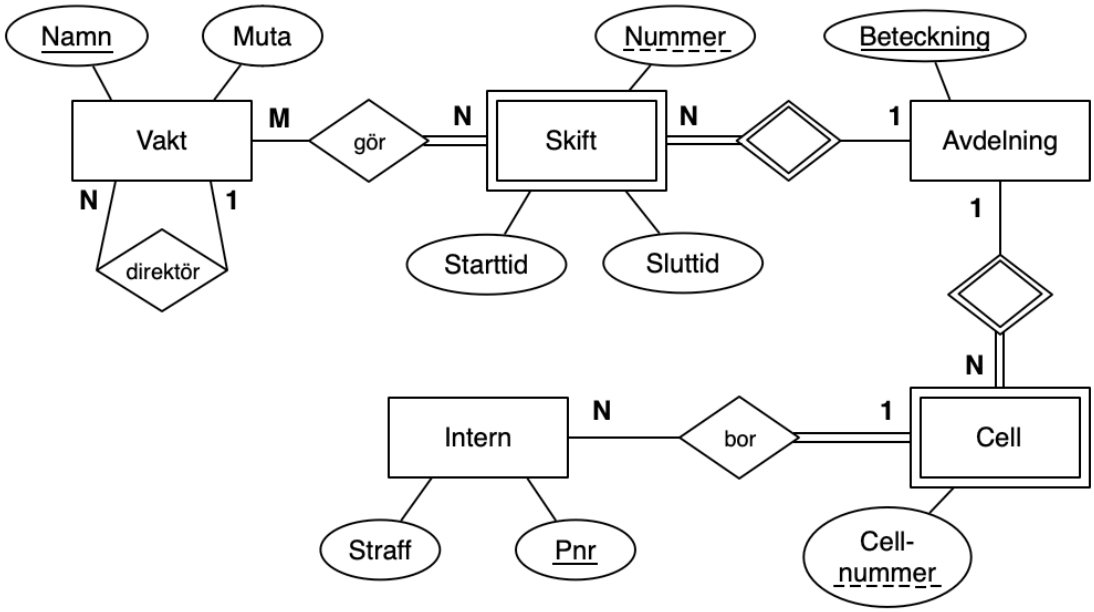
Tävlingsätare (ID , namn , ranking , signaturrätt)
Maträtt (ID , namn , näringsinnehåll)
Tävling (namn , huvudsponsor , rankingpoäng , prispott)
Tour (namn , rykte)
Resultat (ID , prissumma , placering , kommentar , ätar_ID , maträtt_ID , tävlings_namn)
Maträtt_tävling (maträtt_ID , tävlings_namn)
Tävling_tour (tävlings_namn , tour_namn)

Var noga med att inte glömma några främmande nycklar vid relationerna med kardinalitet ”många till en” eller ”en till en”. I detta fall ska signaturrätten ligga som främmande nyckel i tävlingsätare.

Resultat består av 3st identifierande relationer, vilket innebär att 3st olika främmande nycklar måste användas. Detta syns eftersom de tillhörande entiteternas nycklar är främmande med varsitt streck ovanför. Tänk på att ett gemensamt streck över samtliga nycklar är

felaktigt (då antar vi att alla 3 kommer från en och samma entitet, vilket är fel).

13. ER-modell:



En vakt har en mutningssumma, vilket är en egenskap och symboliseras som ett vanligt attribut. En direktör är också en vakt och därför skapas inte en egen entitet för direktör. Istället symboliseras direktören med en vanlig relation med kardinalitet ”många till en” från vakt till sig själv. Detta innebär att en vakt är chef över många vakter. Observera att den ”högsta” vakten är direktör.

Ett skift identifieras med ett nummer tillsammans med en avdelning. Dessutom gäller varje skift för endast en avdelning samt att en avdelning kan ha många skift. Detta innebär att skift måste vara en svag entitet. Den identifierande relationen går till avdelning. Ett skift hade också en starttid som läggs till som ett attribut. Relationen mellan skift och vakt saknade ett streck, vilket lades dit. Observera också att kardinaliteten mellan skift och avdelning har ändrats från ”många till många” till ”många till en”. Alltid vid en identifierande relation används kardinaliteten ”många till en”, det kan aldrig vara något annat.

Avdelningen identifieras av en beteckning och inte ett namn. Varje avdelning består av celler som identifieras med ett cellnummer och till den tillhörande avdelningen. Därför är cell svag entitet. Tyvärr saknas en identifierande relation, vilket det alltid måste finnas. I detta fall går den till avdelning. I en cell måste det finnas minst en eller flera interner, vilket symboliseras med dubbelstreck på relationen.

En intern identifieras av ett personnummer, vilket inte framgick i den felaktiga modellen.

Relationsdatamodell:

Vakt (namn , muta)
Avdelning (beteckning)
Skift (nummer , starttid , sluttid , avdelningsbeteckning)
Cell (cellnummer , avdelningsbeteckning)
Intern (pnr , straff , cellnummer , avdelningsbeteckning)
Gör (vaktnamn , skiftnummer , avdelningsbeteckning)

Svårigheten i denna övning är att få till alla främmande nycklar på ett korrekt sätt. Eftersom kardinaliteten mellan intern och cell är av typen ”många till en” ska nyckel för cell hamna som främmande nyckel i intern. Observera att hela nyckeln måste hämtas, vilket för entiteten cell är cellnummer och avdelningsbeteckning (eftersom cell är en svag entitet). Samma princip gäller för relationen gör. Då gäller det att hämta hela nyckeln för skift. Eftersom skift är svag är nyckeln skiftnummer och avdelningsveteckning, vilket symboliseras med ett heldraget streck ovanför (symboliserar främmande nyckel) och som primärnyckel med streck under.