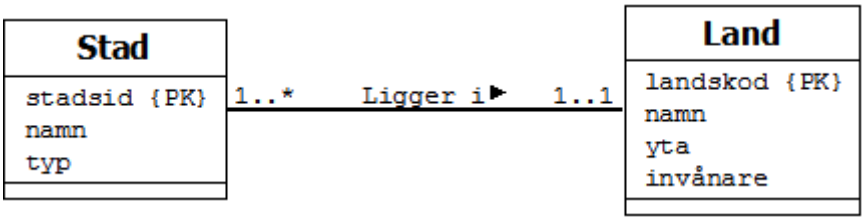
ER-modellering – exempel

Som du känner till från lektion 2 finns det tre typer av samband mellan två relationer (tabeller) i relationsmodellen. Dessa är ett-till-ett, ett-till-många och många-till-många. Dessa sambandstyper finns även i ER-modellen. Vi ska här ta en titt på hur vi i en ER-modell modellerar de olika typerna av samband och hur dessa sen översätts till relationsmodellen på ett korrekt sätt.

Ett-till-många-samband

Denna typ av samband är kanske den enklaste typen av samband att avgöra var främmandenyckeln ska placeras i relationsmodellen. Vid denna typ av samband sätter vi nämligen alltid främmandenyckeln på många-sidan. Ett exempel kan vara sambandet mellan ett land och dess städer. I en ER-modell skulle det kunna se ut så här:



Detta säger oss att en stad "Ligger i" ett och endast ett land (1..1), medan ett land kan ha/innehålla en eller flera städer (1..*). Vi har ett ett-till-många-samband och där Stad är på många-sidan (land har många städer). Här måste vi nu sätta landskod som en främmandenyckel i Stad för att vi ska kunna avgöra vilka städer som ligger i vilka länder. Översatt till relationsmodellen ser det alltså ut så här:

```
Land(landskod, namn, yta, invånare)
Stad(stadsid, namn, typ, landskod*)
```

Notera att vi i ER-modellen inte sätter ut några främmandenycklar utan detta sker först när vi översätter ER-modellen till relationsmodellen. Om vi nu lägger till data i tabellerna kan det se ut så här:

Land

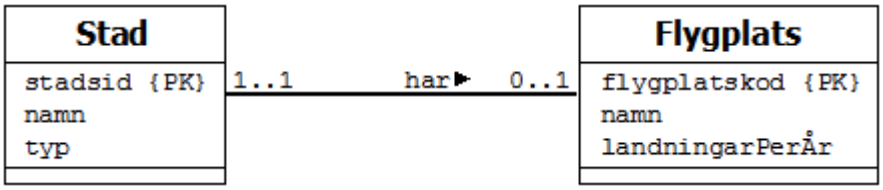
landskod	namn	yta	invånare
SE	Sverige	444964	9453000
NO	Norge	323802	4952000

Stad

stadsid	namn	typ	landskod
001	Oslo	huvudstad	NO
010	Stockholm	huvudstad	SE
532	Östersund	stad	SE
441	Skorped	samhälle	SE
053	Lillehammer	tätort	NO

Ett-till-ett-samband

Vid ett-till-ett-samband spelar det egentligen ingen roll var vi sätter främmande nyckeln, utan vi kan sätta den i valfri tabell som ingår i sambandet. Ett undantag är om ena sidan har deltagandet noll. Ett exempel på detta är stad och flygplats där vi antar att en stad maximalt kan ha en flygplats (vilket inte är riktigt sant). I ER-modellen kan det se ut så här:



Detta säger oss att en stad "har" ingen eller en flygplats (0..1), och att en viss flygplats finns i en och endast en stad (1..1). Vi har ett ett-till-ett-samband, en stad kan ha en flygplats och en flygplats ligger i en stad. Här skulle vi kunna sätta främmande nyckeln i antingen Stad eller i Flygplats, men i och med att Flygplats är på noll-sidan bör vi sätta främmande nyckeln där.

En ok översättning

För att visa varför vi bör sätta främmandenyckeln på just noll-sidan börjar vi med att översätta ER-modellen till relationsmodellen på ”fel” sätt. Det vill säga vi tar flygplatskod och sätter som främmande nyckel i Stad:

```
Stad(stadsid, namn, typ, landskod*, flygplatskod*)
Flygplats(flygplatskod, namn, landningarPerÅr)
```

Lägger vi in lite data kan det se ut så här:

Stad

stadsid	namn	typ	landskod	flygplatskod
001	Oslo	huvudstad	NO	OSL
010	Stockholm	huvudstad	SE	ARN
532	Östersund	stad	SE	OSD
441	Skorped	samhälle	SE	
053	Lillehammer	tätort	NO	

Flygplats

flygplatskod	namn	landningarPerÅr
OSL	Gardermoen	115211
ARN	Arlanda	106428
OSD	Åre Östersund	3977

Som du ser i tabellen Stad ovan saknas det för några rader i kolumnen flygplatskod ett värde. Dessa innehåller i stället ett så kallat NULL-värde. I det exempel vi har ovan är det endast två städer som saknar flygplats, men tänker vi oss att vi i databasen lägger in alla länders städer förstår vi att det innebär väldigt många rader med NULL-värden. Om möjligt ska vi försöka designa databasen så att dessa NULL-värden undviks.

En bättre översättning

Eftersom en flygplats alltid finns i en stad undviker vi NULL-värden om vi i stället tar stadsid och sätter som en främmande nyckel i Flygplats:

```
Stad(stadsid, namn, typ, landskod*)
Flygplats(flygplatskod, namn, landningarPerÅr, stadsid*)
```

Lägger vi in lite data kan det se ut så här:

Stad

stadsid	namn	typ	landskod
001	Oslo	huvudstad	NO
010	Stockholm	huvudstad	SE
532	Östersund	stad	SE
441	Skorped	samhälle	SE
053	Lillehammer	tätort	NO

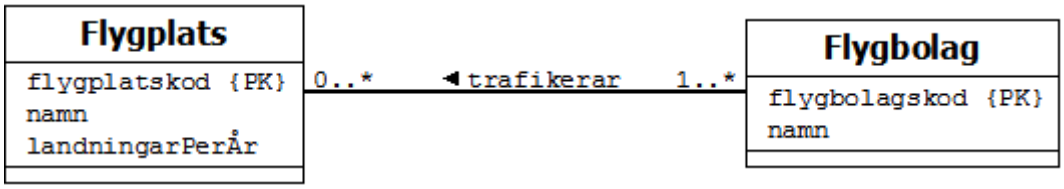
Flygplats

flygplatskod	namn	landningarPerÅr	stadsid
OSL	Gardermoen	115211	001
ARN	Arlanda	106428	010
OSD	Åre Östersund	3977	532

Många-till-många-samband

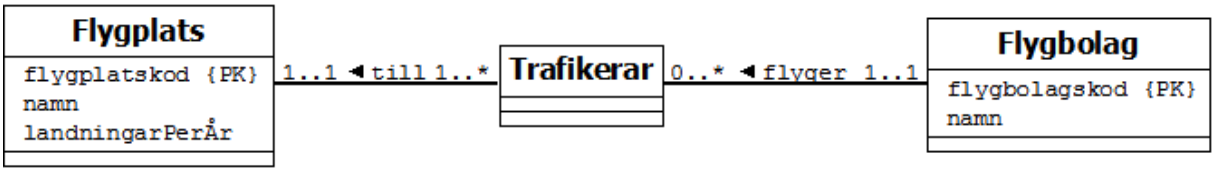
Sista sambandet, många-till-många, är lite speciellt då vi inte direkt kan placera ut några främmande nycklar. Det vi alltid måste göra vid många-till-många-samband är använda en extra tabell som en koppling mellan de båda ursprungstabellerna. Denna "kopplingstabell" kommer att innehålla två attribut, nämligen de båda primärnycklarna från ursprungstabellerna. Dessa kommer att bli en sammansattnyckel i kopplingstabellen och även främmande nycklar till ursprungstabellerna.

Ett exempel är flygplats och flygbolog. En flygplats kan trafikeras av många olika flygbolog, och ett visst flygbolog kan trafikera många olika flygplatser. I ER-modellen kan det se ut som följer:



Detta säger oss att ett flygbolag "trafikerar" ingen, ett eller flera flygbolag (0..*) och att en flygplats trafikeras av ett eller flera flygbolag (1..*). Här kan vi inte ta flygbolagskod och sätta som främmande nyckel i Flygplats. Inte heller kan vi ta flygplatskod och sätta som främmande nyckel i Flygbolag (eller för den delen sätta främmande nycklar i båda tabellerna). Varför? Detta lämnar vi som en övning åt dig att ta reda på (prova att skapa tabeller och fyll dem med exempeldata).

Vid många-till-många-samband måste vi alltid skapa en ny relation i relationsmodellen där vi sätter våra främmande nycklar. Om vi skulle lägga till den nya relationen som en entitetstyp i ER-modellen, vilket inte är ett krav när vi använder UML, får vi detta:



Detta kallas för att objektifiera en sambandstyp. Sambandstypen "trafikerar" blir då en egen (svag) entitetstyp med nya sambandstyper till de ursprungliga entitetstyperna. De nya sambandstyperna är alltid av typen ett-till-många där den nya entitetstypen är på många-sidan. Vi sätter alltid 1..1 vid de ursprungliga entitetstyperna eftersom varje förekomst (eller rad) i Trafikerar gäller för en viss kombination av flygplats och flygbolag. Ett flygbolag kan aldrig trafikera samma flygplats många gånger.

Oavsett hur vi väljer att modellera i ER-modellen (båda exemplen ovan innebär samma sak) översätter vi det till relationsmodellen så här:

```
Flygplats(flygplatskod, namn, landningarPerÅr, stadsid*)
Trafikerar(flygplatskod*, flygbolagskod*)
Flygbolag(flygbolagskod, namn)
```

De båda primärnycklarna från ursprungstabellen bildar en sammansattnyckel i "kopplingstabellen" Trafikerar. Vi sätter även dessa båda till främmande nyckel så att vi kan avgöra vilket flygbolag som trafikerar vilken flygplats, och tvärtom. För att lättare se hur detta fungerar lägger vi in lite data i tabellerna:

Flygplats

flygplatskod	namn	landningarPerÅr	stadsid
OSL	Gardermoen	115211	001
ARN	Arlanda	106428	010
OSD	Åre Östersund	3977	532

Trafikerar

flygplatskod	flygbolagskod
ARN	SAS
ARN	NAX
OSL	NAX
OSD	SAS
OSL	SAS

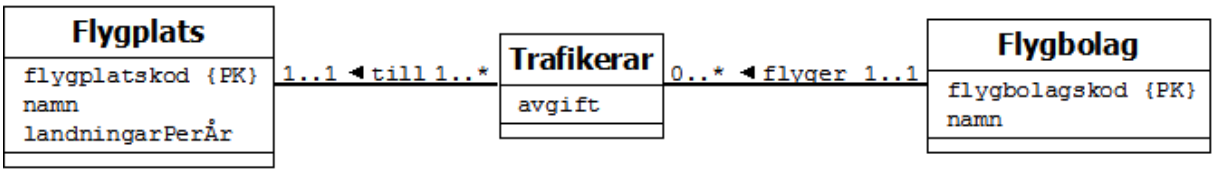
Flygbolag

flygbolagskod	namn
SAS	Scandinavian Airlines
NAX	Norwegian Air Shuttle
SCW	Malmö Aviation

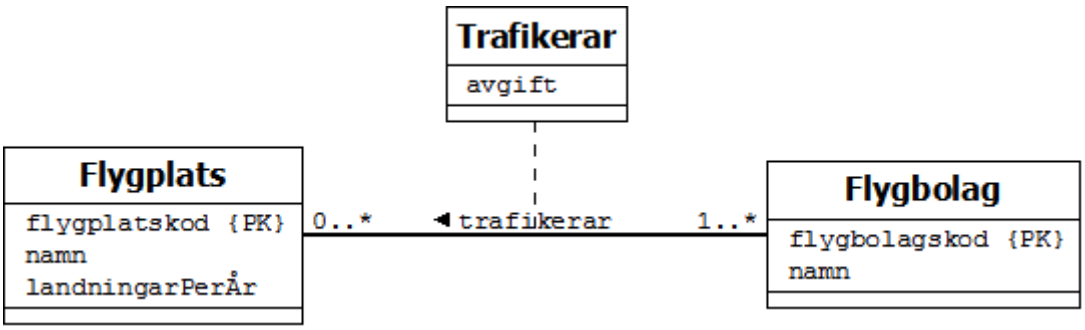
I Trafikerar fyller vi i flygplatskoden för vilken flygplats det gäller, samt flygbolagskoden för det flygbolag som trafikerar den flygplatsen. Detta får vi göra för alla kombinationer av flygplatser och flygbolag. Från tabellen kan vi se att det enbart är SAS som trafikerar alla (tre) flygplatser. Flygbolagskoden SAS återfinns för alla flygplatskoder. Vi ser även att Malmö Aviation inte trafikerar någon flygplats alls. Dess flybolagskod SCW återfinns inte som värde i flygbolagskod i tabellen Trafikerar.

Sambandstyper med egna attribut

Ibland kan det vara nödvändigt att låta sambandstyper ha egna attribut. I det exempel vi tittat på här kan vi tänka oss att flygbolagen måste betala en landningsavgift varje gång de landar på en flygplats. Denna landningsavgift kan för ett flygbolag variera från flygplats till flygplats. Detta kan vi implementera genom att i ER-modellen låta sambandstypen "trafikerar" ha ett attribut med namnet avgift. I UML kan vi antingen objektifiera sambandstypen och låta avgift vara ett attribut i den nya entitetstypen:



Vi kan även använda oss av en så kallad associationsklass i vilken vi lägger in attributet (se dokumentet om Dia Diagram Editor för hur detta görs):



Oavsett vilket av sätten vi väljer översätts det till relationsmodellen på samma sätt:

```
Flygplats(flygplatskod, namn, landningarPerÅr, stadsid*)
Trafikerar(flygplatskod*, flygbolagskod*, avgift)
Flygbolag(flygbolagskod, namn)
```

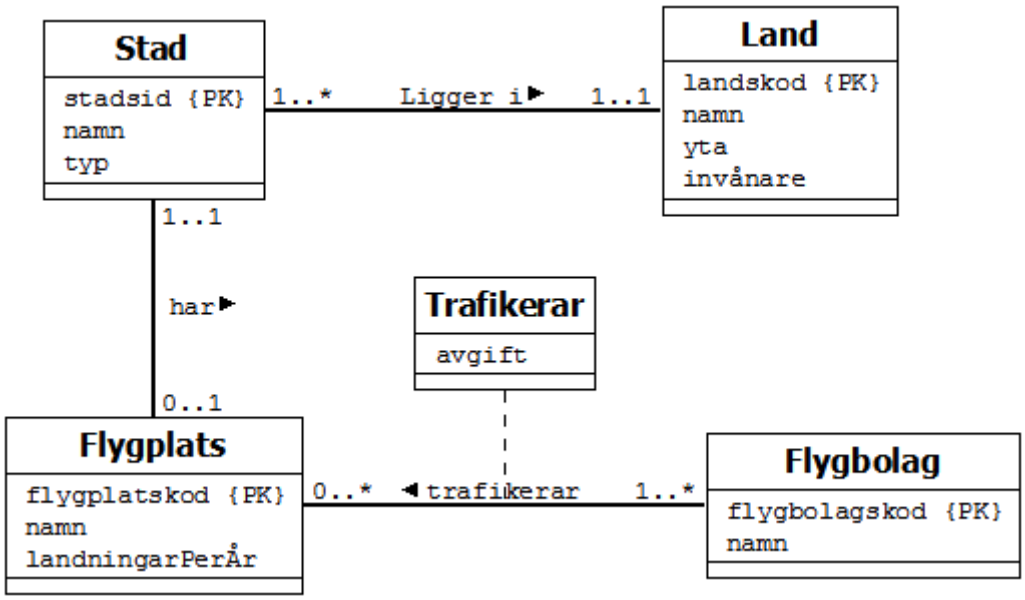
Med lite data inlagt i tabellen Trafikerar kan det se ut så här:

Trafikerar

flygplatskod	flygbolagskod	avgift
ARN	SAS	1000
ARN	NAX	1500
OSL	NAX	1200
OSD	SAS	500
OSL	SAS	2000

Flygbolaget Scandinavian Airlines (SAS) får till exempel betala 1000 för att landa på Arlanda, medan det bara kostar 500 för SAS att landa på Åre Östersund.

Från alla exemplen ovan får vi följande kompletta ER-modell:



Som översatt till relationsmodellen ser ut så här:

```
Land(landskod, namn, yta, invånare)
Stad(stadsid, namn, typ, landskod*)
Flygplats(flygplatskod, namn, landningarPerÅr, stadsid*)
Trafikerar(flygplatskod*, flygbolagskod*, avgift)
Flygbolag(flygbolagskod, namn)
```

Förhoppningsvis har detta exempel gett dig en bild hur vi modellerar de olika typerna av samband och framför allt hur de översätts till relationsmodellen. I inlämningsuppgift 3 och 4 kommer du själv att få prova på att skapa ER-modellen och översätta dessa till relationsmodellen.