ER modellering

Modul 9 (uke 7) innhold

SQL-scripts (.sql filer)

- Hittil har vi sett på hvordan det er å skrive SQL mot eksisterende databaser. Vi har hatt fokus på:
  - select queries.
  - create table, alter table og drop table.
  - insert into, update og delete from.

- NB: Vi kan lagre alle typer SQL statements for senere kjøring.
  - I praksis vil de lagres i en tekstfil, som vi gir .sql endingen.
  - Se world schema for eksempel/detaljer.
  - *Merk: Dere skal levere SQL som scripts på kommende checkpoint! ;-)*

Modellering

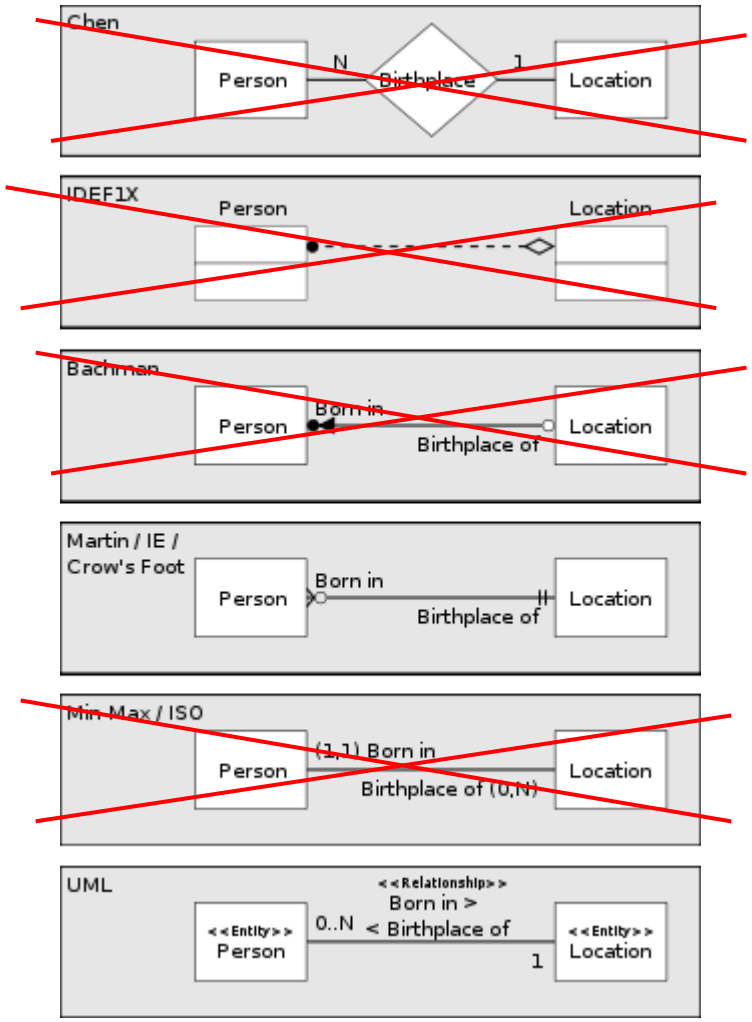
- Av og til ønsker vi å være arkitektene som designer databasen.
  - Da kommer ER modellering inn i bildet!

- ER modellering: (noen kaller det EAR modellering)
  - ER = Entity Relationship (EAR = Entity *Attribute* Relationship)

- NB: Uttrykk – her må vi holde tunga rett i munnen:
  - "Relation" er et generelt relasjonsdatabaseuttrykk, og betyr tabell.
  - "Relationship" brukes i modelleringssammenheng, og betyr koplingen mellom to tabeller.
  - På norsk kan vi oversette *relationship* til *forhold*.

Om notasjoner

- Det finnes en rekke ER notasjoner.
- Vi skal benytte UML.
- En annen vanlig variant er Kråkefot.
- Resten er mindre brukt.



(Kilde: Wikipedia)

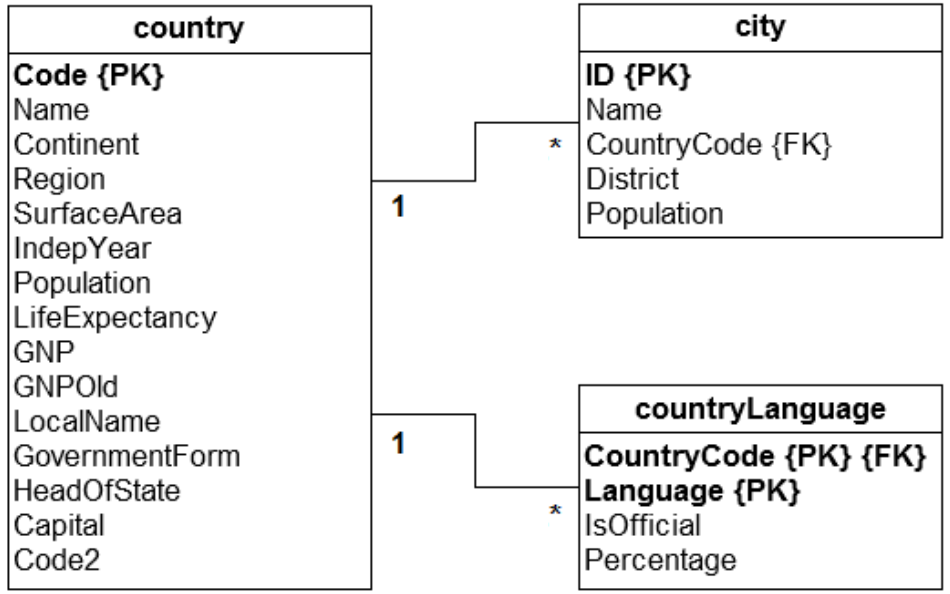
ER diagram (modellering)

- Vi kan vise modellen av databasen *World* som et ER diagram:
  - Entity
  - Relationship

- Denne modellen er laget i Gliffy: www.gliffy.com/

- Lucidchart og Draw er gode alternativer.

- Velg selv, gjerne noe som er gratis? :-P



Eksempel case: prosjektstyring

- Et firma ønsker å få oversikt over sine prosjekter. De har leid oss inn for å lage en databaseløsning som ordner dette.

- De ønsker spesifikt å få oversikt over følgende:
  - Hvilken avdeling (nummer, navn) eier hvert prosjekt?
  - Hvilke prosjekter (nummer, tittel) involverer hvilke ansatte (id, navn)?

- Modelleringsspørsmål:
  - Hvilke entiteter (kommende tabeller) må vi ha?
  - Hvilke attributter skal plasseres i entitetene?
  - Hva er forholdene mellom entitetene? (Hvordan henger de sammen?)

Entiteter og attributter

- Ut fra spesifikasjonen kommer vi fram til et behov for følgende entiteter:
  - Avdeling
  - Prosjekt
  - Ansatt

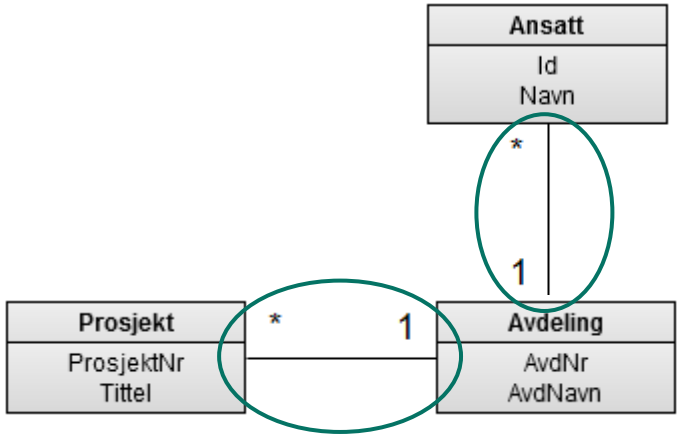
- Videre trenger vi følgende attributter:
  - Avdeling: AvdNr, AvdNavn
  - Prosjekt: ProsjektNr, Tittel
  - Ansatt: Id, Navn

Forhold

- Forholdene er ikke oppgitt i spesifikasjonen, men det er logisk(?) å anta forhold som oppgitt under.
  - (Vi bør tidlig i prosjektet få de bekreftet av kunden!)

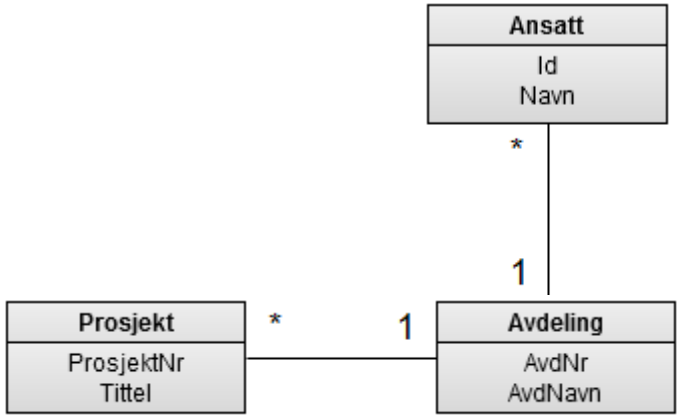
- Forholdet avdeling & ansatt:
  - En avdeling kan ha mange (symbol: *) ansatte.
  - En ansatt tilhører én (symbol: 1) avdeling.

- Forholdet prosjekt & avdeling:
  - Et prosjekt tilhører én (symbol: 1) avdeling.
  - En avdeling kan ha mange (symbol: *) prosjekter.



Eksempel case: prosjektstyring, ekstra innhold

- Modellen begynner å falle på plass! Men firmaet har mer info de vil registrere:
  - Hvor mye tid benytter hver ansatt (id, navn) per prosjekt?

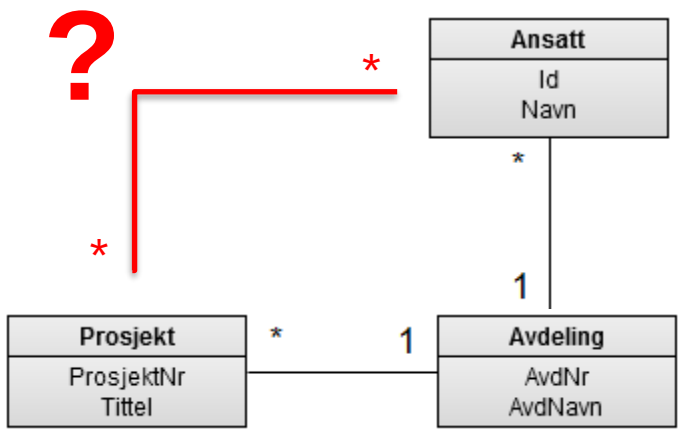


- (Denne er litt vanskelig å plassere på rett sted?)

Forhold – forts.

Vi mangler 1 forhold:

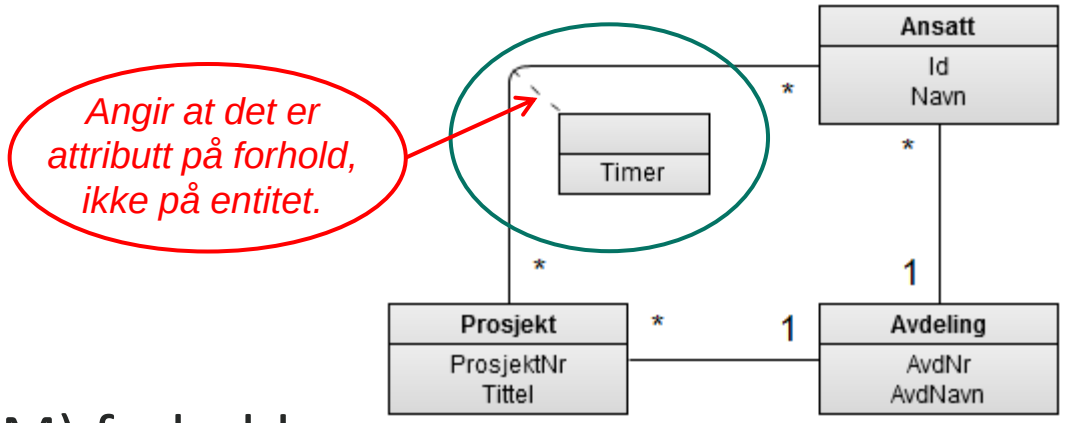
- Forholdet prosjekt & ansatt. Vi antar:
  - Et prosjekt kan bemannes av mange (*) ansatte.
  - En ansatt kan parallelt jobbe på mange (*) prosjekter.



- Dette er et M:M forhold! Men hvordan kan vi legge til det?

M:M forhold

- Forholdet prosjekt & ansatt:
  - Et prosjekt kan bemannes av mange (*) ansatte.
  - En ansatt kan jobbe parallelt på mange (*) prosjekter.



- Altså et mange-til-mange (M:M) forhold.
  - (Kommer tilbake til hvordan vi lager det i DB.)

- Og vi har én attributt igjen:
  - Vi trenger å vite ”tid per ansatt per prosjekt”.
  - Men putte denne hvor?
  - På forholdet prosjekt & ansatt!

Koblingsentiteter

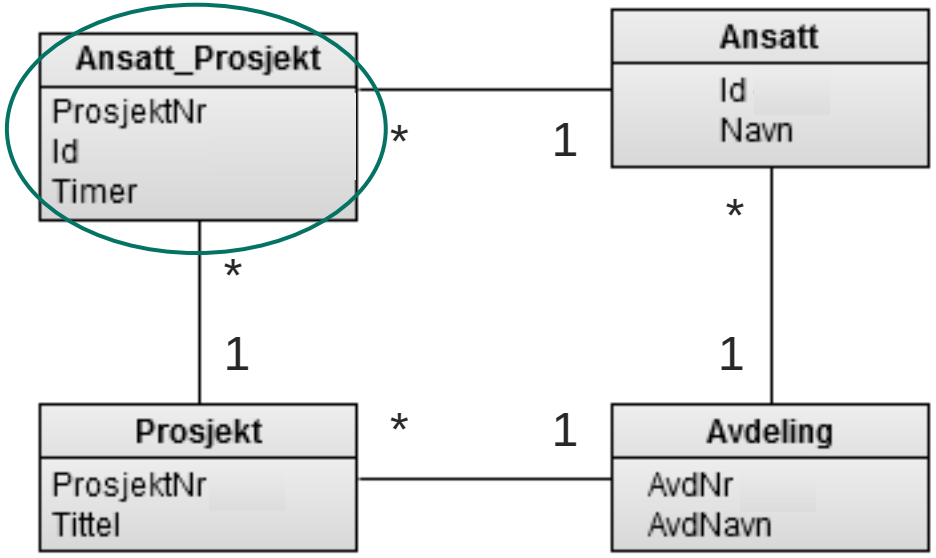
- Vi la inn et attributt på forholdet mellom Ansatt og Prosjekt.
  - Forholdet er i dette tilfellet et mange-til-mange-forhold: En ansatt kan delta i mange prosjekter. Et prosjekt kan ha mange prosjektdeltakere (ansatte).

- Når vi har et mange-til-mange-forhold introduserer vi en koblingsentitet ("koblingstabell"), og relevante attributter ("kolonner") legges i denne entiteten.

Fra ER-modell til fungerende database

For å gå fra ER-modell til fungerende database må vi gjennom 3 steg:

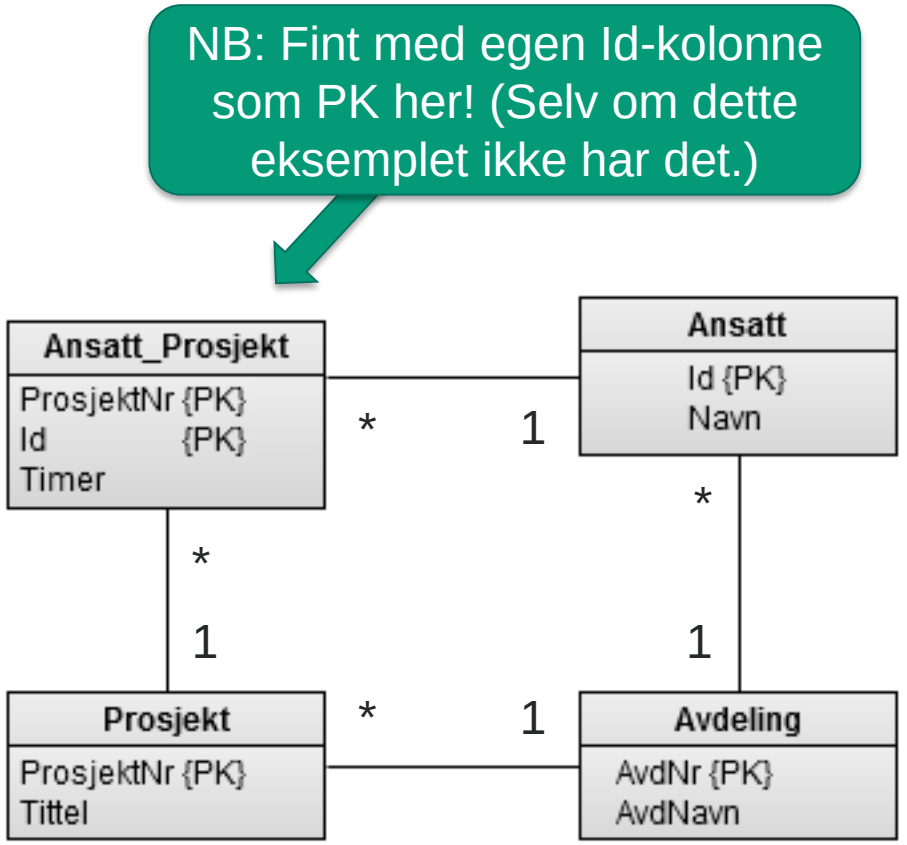
1. Fjern mange-til-mange forhold, ved å lage koplingsentiteter.
   - Koplingsentitetene må inneholde PK fra de to entitetene de kopler sammen.
   - Og vi legger inn eventuelle forholds-attributter ("Timer" i vårt tilfelle).



Fra ER-modell til fungerende database – forts.

2. Hver entitet blir en tabell med samme navn.
   – (Attributtene blir kolonner.)

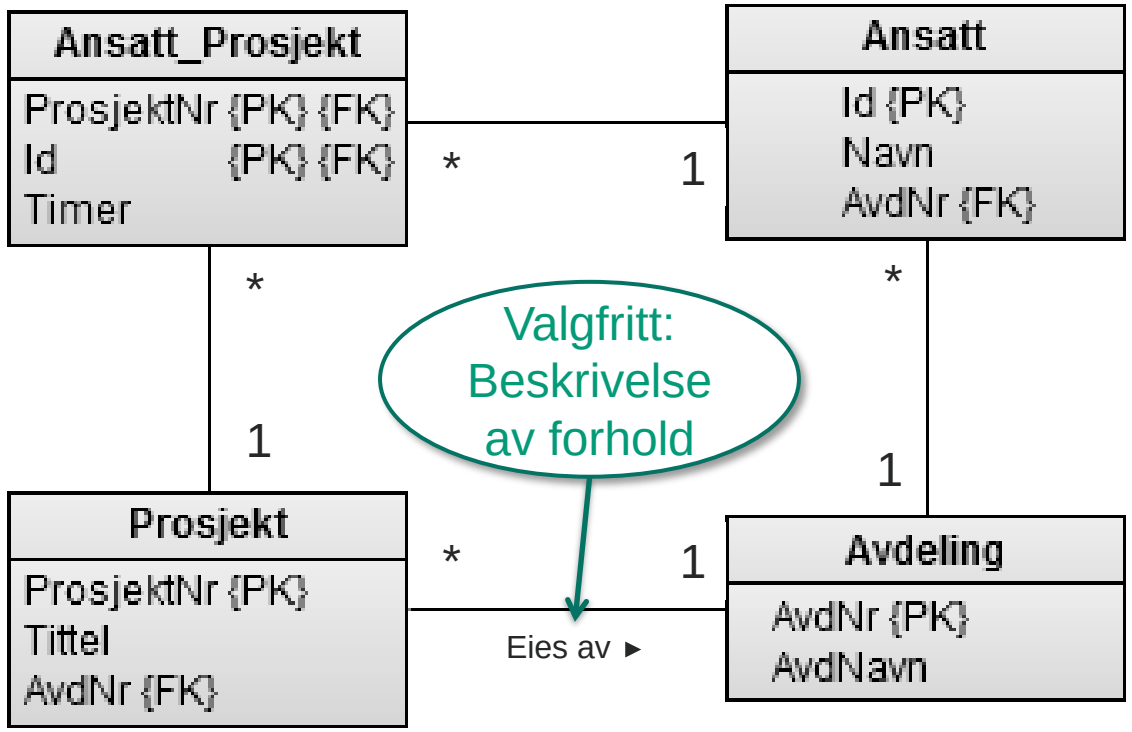
- Vi angir Primary Key for alle tabeller.
  – I UML er det vanlig å angi primary key ved å sette {PK} bak navnet.
  – (Underline er en annen, vanlig primary key notasjon.)



Fra ER-modell til fungerende database – forts.

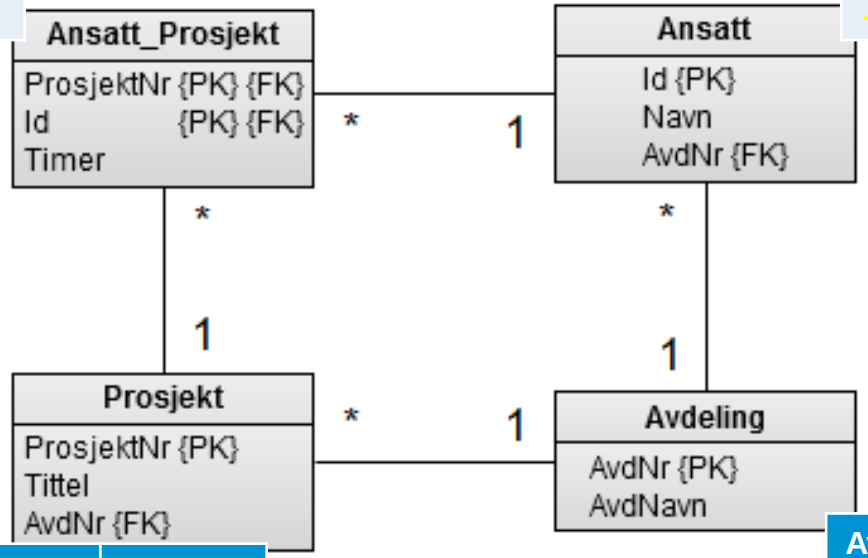
3. Legg til Foreign Keys.
   - Én til mange forhold realiseres ved at PK på én-siden kopieres til mange-siden, som fremmed-nøkkel der.
   - UML notasjon, foreign key: {FK}.

- Om vi vil, setter vi navn (og "retning for navnet") på forholdene.



ProsjektNr	Id	Timer
4	11	5
2	13	6
3	14	12
4	13	17

Id	Navn	AvdNr
11	Jon	111
12	Ida	113
13	Ole	111
14	Eli	114



ProsjektNr	Tittel	AvdNr
1	Grindgut	111
2	Apollo	113
3	Ulv	113
4	Follobanen	111

AvdNr	AvdNavn
111	Oslo
112	Bergen
113	Mandal
114	Bodø

Videre i dag

- Nå:
  - Øvinger (modellering, 2 oppgavesett)

- Avslutning i dag, litt teori ifbm. modellering:
  - Databaseuttrykk ifbm. forhold
  - Surrogatnøkler
  - Bittelitt om GDPR


Academy