Notater: INF1300

Veronika Heimsbakk
veronahe@student.matnat.uio.no

29. mai 2013

Innhold

1 ORM

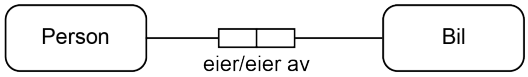
1.1 Setningers aritet

Den elementære setningen *Studenten med navn Anne fikk i emned med emnekode INF1010 resultatet karakteren B* inneholder tre begreper: *student*, *emne* of *resultat*. Antall begreper i en setningen kalles setningens aritet. Vårt eksempel har aritet 3.

- Setninger med aritet 1 kaller vi unære.
- Setninger med aritet 2 kaller vi binære.
- Setninger med aritet 3 kaller vi ternære.

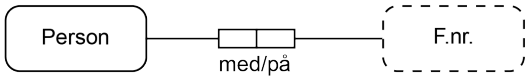
Setninger med aritet større enn 3 er sjeldne.

1.2 Faktatyper og broer i ORM



Figur 1: Et eksempel på faktatype mellom begrepen Person og Bil.

En **bro** er en forbindelse mellom et begrep og en representasjon.

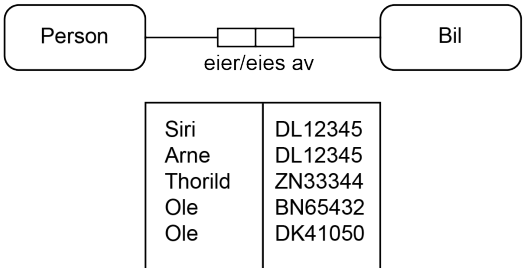


Figur 2: Et eksempel på en bro.

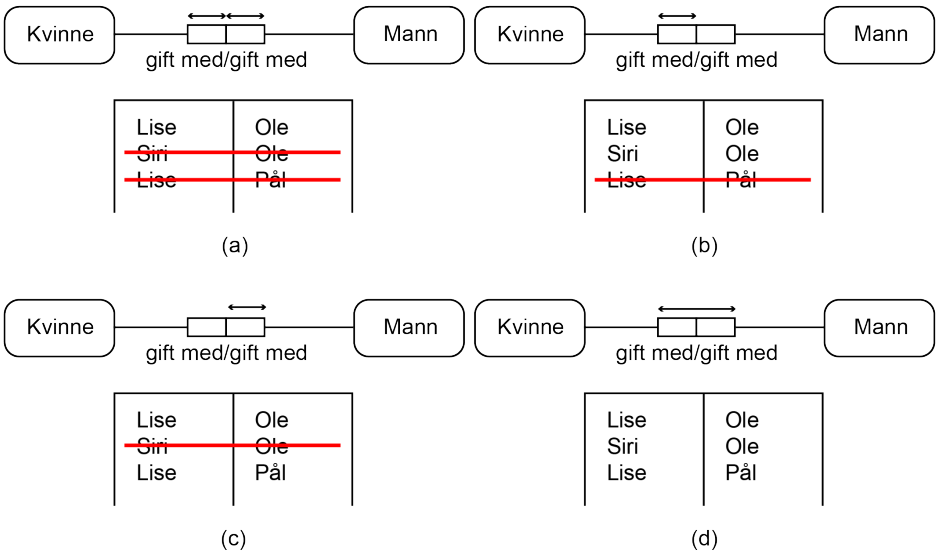
Setningstyper er en fellesbetegnelse på faktatyper og broer. Broer er alltid binære, faktatyper kan ha vilkårlig antall roller (aritet). I faktatyper bør alle rollenavn inneholde er verb. I alle broer er det vanlig med preposisjoner som rollenavn. De to vanligste rolleparene er **med/for** og **med/på**.

1.3 Forekomsttabeller og entydighetsskranker

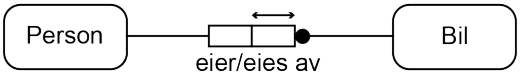
I Fig. 4(a) er Norges eneste lovlige ekteskapsform, denne kalles 1:1 (én-til-én) faktatype mellom (bekrepene) Kvinne og Mann. I Fig. 4(b) har vi flerkoneri, dette er n:1 (mange-til-én). I Fig. 4(c) har vi flermanneri, dette er 1:n (én-til-mange). I Fig. 4(d) har vi flergifte, dette er m:n (mange-til-mange) faktatype.



Figur 3: Eksempel på forekomsttabell.



Figur 4: Eksempel på entydighetsskranker.

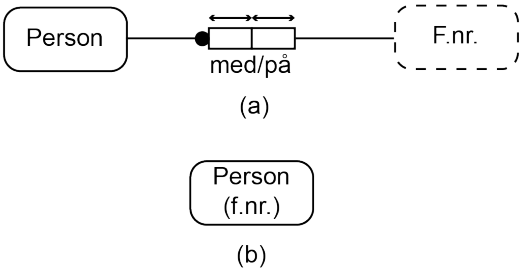


Figur 5: Eksempel på en total rolle.

1.4 Totale roller og perfekte broer

Dersom alle biler har en eier, sier man at vi har en total funksjon fra Bil til Person (den er definert for alle forekomster av Bil). Vi sier at rollen *eies av* er en **total rolle** for Bil og markerer det med en kule (liten fylt sirkel) på rollen.

En 1:1 bro der begrepsrollen er total, kalles en **perfekt bro**. Perfekte broer (Fig. 5(a)) er så vanlige at vi har en egen kortform for dem, vist i Fig.

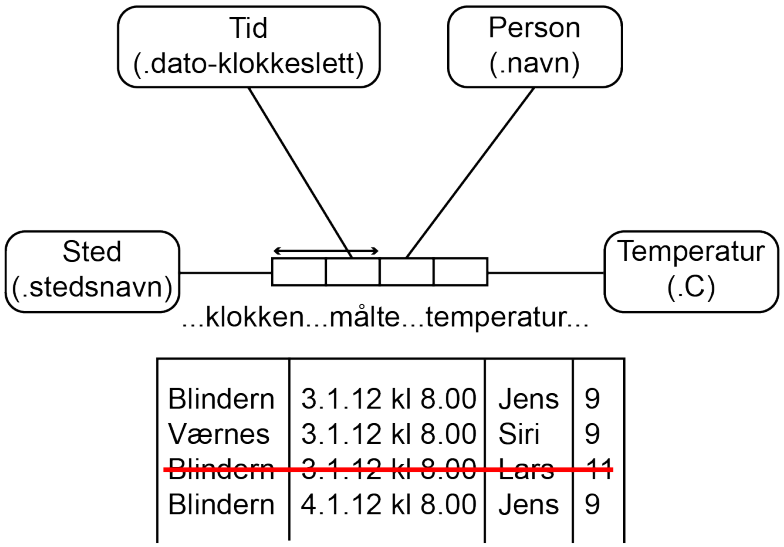


Figur 6: Eksempel på en perfekt bro.

5(b). De to tegnemåtene er ekvivalente.

1.5 Begrepsdannelse

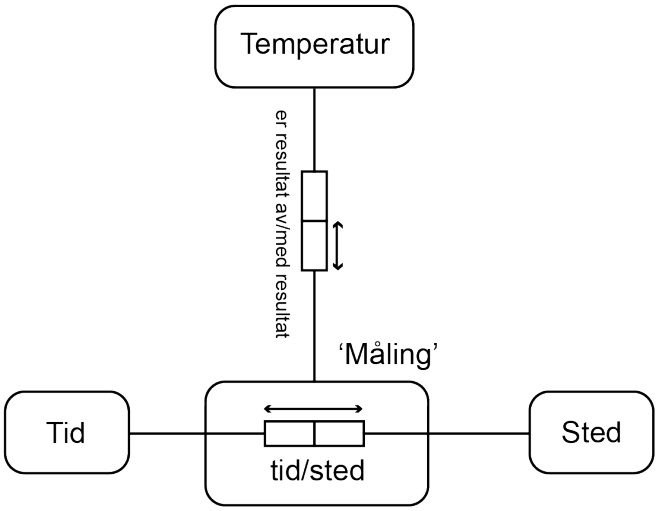
Setning: *På Bildern klokken 8 målte Jens 9 grader.* Her er *Blindern* et **sted** med representasjon **stedsnavn**. *8* er et **tidspunkt** med representasjon **dato og klokkeslett**. *Jens* er en **person** med representasjon **personnavn**. *9* er en **temperatur** med representasjon **°C**



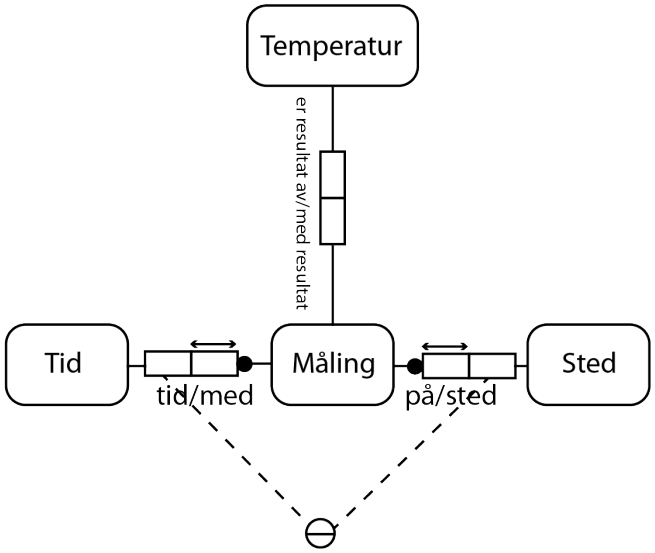
Figur 7: Eksempel på begrepsdannelse.

1.6 Ekstern entydighetsskranke

Entydighet på tvers av faktatyper indikerer vi med en **ekstern entydighetsskranke** på de involverte rollene *tid* og *sted*. Fig. 9.



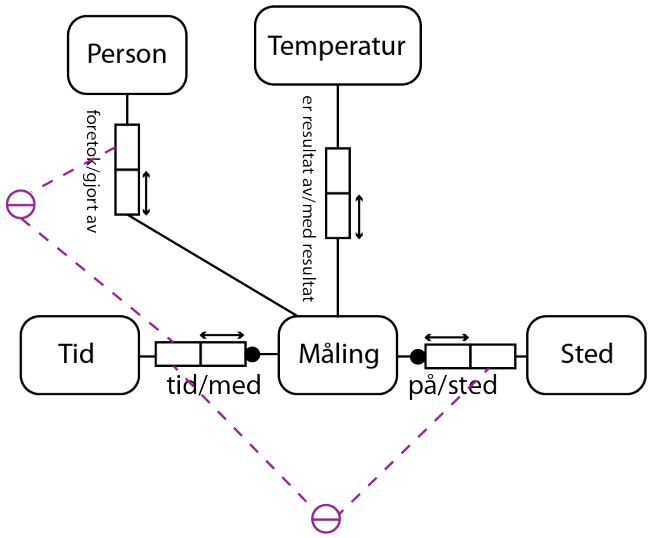
Figur 8: Alternativ notasjon.



Figur 9: Eksempel på en ekstern endydighetsskranke.

I Fig. 10, uttrykker vi at en person ikke kan foreta mer enn én måling av gangen. Dette gjøres med en ekstern entydighetsskranke på rollene *foretok* og *tid*.

Alle entydighetsskranker som går over mer enn én rolle i en faktatype, skjuøer et (nytt) begrep. Man skal alltid vurdere om det skal lages nye begreper når man får faktatyper med lange entydighetspiler. En faktatype med aritet 3 eller 4 (eller mer) kan gjøres om til binære setninger ved å lage



Figur 10: Eksempel på en ekstern endydighetsskranke.

ett eller flere nye begreper. Eksempelsvis har Fig. 7 aritet 4.

1.7 Populasjoner

Populasjon for en rolle Hvis r er en rolle, betegner **pop(r)** mengden av forekomster i kolonnen for r i forekomsttabellen.

Populasjon for et begrep Begreper har egentlig ikke forekomster løsrevet fra roller, men vi definerer likevel populasjonen til et begrep A som har roller $r_1, r_2, \ldots, r_n$ ved $pop(A) = pop(r_1) \cup pop(r_2) \cup \cdots \cup pop(r_n)$

Merk: populasjonen til en rolle/et begrep varierer med tiden.

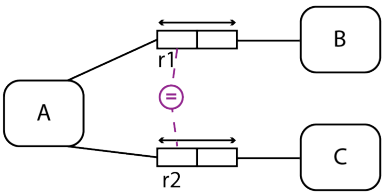
1.8 Mengdeskranker

Mengdeskrankene begrenser mengden av forekomster i en eller flere roller i forhold til forekomstene i andre roller. Det finnes følgende varienter:

- Likhetsskranke
- Ulikhetsskranke
- Delmengdeskranke

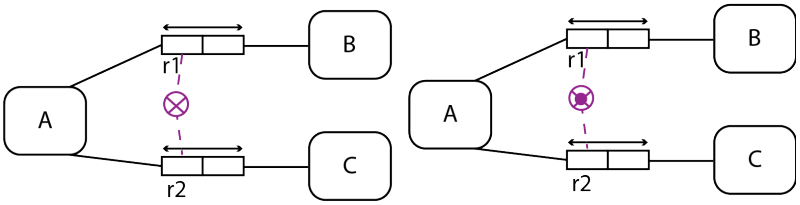
1.8.1 Mengdeliketsskranke

A skal ha rollen $r1$, hvis og bare hvis, A har rollen $r2$. $pop(r1) = pop(r2)$ for alle tilstander.



Figur 11: Eksempel på en ekstern mengdelikhetsskranke.

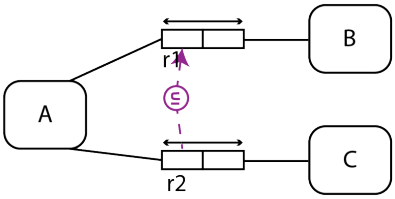
1.8.2 Mengdeulikhetsskranke



Figur 12: Eksempel på en ekstern mengdeulikhetsskranke.

A skal ikke ha både rollen $r1$ og $r2$. $pop(r1) \cap pop(r2) = \emptyset$
I den andre figuren i Fig. 12, skal A ha en og bare en av rolle $r1$ og $r2$.

1.8.3 Delmengdeskranke



Figur 13: Eksempel på en ekstern delmengdeskranke.

Hvis A har rollen $r2$, så skal A også ha rollen $r1$. $pop(r2) \subseteq pop(r1)$ for alle tilstander.

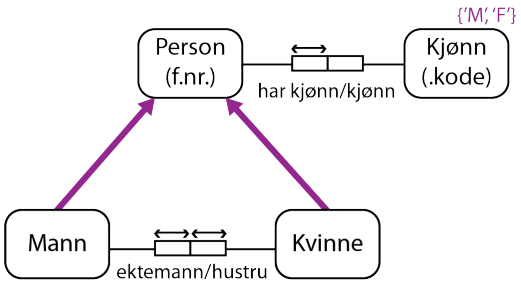
1.9 Underbegreper

Kan alle tenkelige forekomster av et begrep spille alle roller som er knyttet til begrepet? Hvis nei, kan vi få en mer presis modell ved å innføre **underbegreper**.

B er et underbegrep av A hvis vi alltid har at $pop(B) \subseteq pop(A)$.

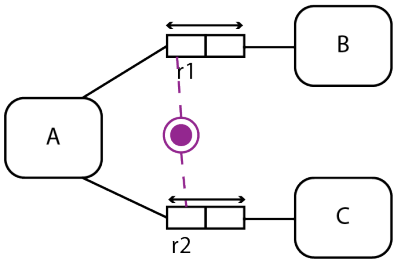
Figur 14: Notasjon: underbegreper.

Underbegreper arver representasjon og roller fra superbegrepet. I tillegg har de sine egne roller. Underbegrepsskranker brukes til å bestemme hvilket underbegrep hver enkelt forekomst tilhører. De kan overlappe eller være disjunkte. Underbegreper kan, men må ikke, være uttømmende mhp. sitt superbegrep. Resonnementer over entydighetsskranker, totale rolle og underbegrepsskrankene avslører om underbegrepene er overlappende og/eller uttømmende.



Figur 15: Eksempel med underbegrepsskranke. Hver Mann er en Person som har *kjønn* 'M'. Hver Kvinne er en Person som har *kjønn* 'F'.

1.10 Kombinert total rolle



Figur 16: Kombinert total rolle.

A skal ha enten rollen $r1$ eller rollen $r2$. $pop(r1) \cup pop(r2) = pop(A)$ for alle tilstander.

2 SQL

2.1 Nøkler og nøkkelatributter

Figur 17: To ansatte skal ikke kunne ha samme Ans# og to personer kan aldri ha samme fødselsnummer.

Primærnøkler blir markert med én strek. Andre kandidatnøkler er i dette tilfellet markert med to streker.

2.1.1 Fremmednøkler

Fremmednøkler må ha samme antall attributter som primærnøkkelen i den relasjonen den peker ut, og attributtene må ha parvis samme domener. (Noen databasesystemer tillater også fremmednøkler til kandidatnøkler som ikke er primærnøkler.) Korresponserende attributter behøver ikke å ha samme navn. Det er lov å ha fremmednøkler til «seg selv». Fremmednøkler benyttes til å uttrykke integritetsregler.

Påkrevde integritetsregler i relasjonsdatabaser

- **Entitetsintegritet:** Alle relasjonsskjemaer skal ha en og bare en primærnøkkel. Ingen av attributtene i en primærnøkkel får være **null**.
- **Referanseintegritet:** Hvis fremmednøkkelen ikke er **null**, så skal det finnes et tuppel i den refererte relasjonen hvor primærnøkkelen har samme verdi som fremmednøkkelen (dvs. at det refererte tuppelet skal eksistere).

2.2 create table

```
1  Ansatt(Aid, Navn, Tittel, Fdato, Pnr, AnsDato)
2
3  create table Ansatt (
4    Aid int primary key,
5    Navn  varchar(40) not null,
6    Tittel  varchar(15) not null,
7    Fdato char(6) not null,
8    Pnr char(5) not null,
9    AnsDato date,
10 unique (Fdato, Pnr)
11 );
```

code/create.sql

Ansatte skal ikke kunne ha samme AId, to personer kan aldri ha samme fødselsnummer = Fdato + Pnr. Dermed er både AId of (Fdato, Pnr) kandidatnøkler. Vi velger AId som primærnøkkel og markerer (Fdato, Pnr) som kandidatnøkkel.

Maks en primærnøkkeldeklarasjon pr. relasjon. Kandidatnøkler kan deklareres i `create table` sammen med nøkkelattributtet med `unique`.

2.3 Skranke på ett attributt

```
1 create table Ansatt (... Fdato int not null, ..);
```

code/notnull.sql

Kan ikke sette inn tuppel med verdien null i dette attributtet. Kan ikke endre verdien til null senere.

```
1 create table Ansatt (
2   ...
3   Tittel varchar(15)
4   check (Tittel='Selger' or Tittel='Direktor' or ...),
5   ...);
```

code/check.sql

Angir en betingelse på attributtet. Sjekkes ved hver endring av attributtets verdi.

2.4 select

```
1 select    Pnavn, StartDato
2 from    Kunde k, Prosjekt p
3 where   Knavn = 'Pust og pes AS' and k.KId = p.KId
4 order by  StartDato desc;
```

code/select.sql

Svar på oppgaven: finn navn og startdato for alle prosjekter bestilt av kunden «Pust og pes AS». Sorter dem slik at det nyeste prosjektet kommer først.

2.5 Tekstmønstre

```
1 select firstname from person
2 where firstname like 'O_a';
3
4 select firstname from person
5 where firstname like 'O%a';
```

code/tekst.sql

_ passer med *ett* vilkårlig tegn, og % passer med en vilkårlig tekststring (flere enn ett tegn).

```
1 select firstname || ' ' || lastname as navn, gender as
    kjonn
2 from person
```

```
3 where firstname like '___' and lastname not like '%sen';
```

code/tekst2.sql

Her blir resultatet en tabell over navn og kjønn på personer som har akkurat tre tegn i fornavn og et etternavn som ikke slutter på *sen*.

2.6 Aggregeringsfunksjoner

navn	virkning
count	teller antall
min	finner minste verdi
max	finner største verdi
sum	summerer verdier
avg	finner snittet av verdier

Tabell 1: Aggregeringsfunksjoner.

2.6.1 count()

```
1 select count(*) from person;
2
3 select count(*) as antTupler from person;
4
5 select count(gender) from person;
6
7 select count(distinct firstname) from person;
```

code/count.sql

1. Gir antall tupler i tabellen.
2. Som for alle attributter i select-listen, kan vi gi `count(*)` et nytt navn.
3. Gir antall tupler i tabellen hvor attributtet `gender` ikke er null.
4. Gir antall forskjellige verdier i attributtet `firstname` (null telles ikke med).

2.6.2 min() og max()

```
1 select max(lonn) - min(lonn)
2 from ansatt
3 where avd = 'ifi';
```

code/minmax.sql

Ved tabellen `ansatt(anr, navn, lonn, avd)`, så finner vi her den største lønnsforskjellen ved Ifi. Det er ikke lov å ha regneuttrykk som parameter i `min()` og `max()`.

2.6.3 sum() og avg()

```
1 select sum(lonn), avg(lonn)
2 from ansatt
3 where avd = 'ifi';
```

code/sumavg.sql

Ved tabellen `ansatt(anr, navn, lonn, avd)`, så finner vi her summen av lønnsutgifter og gjennomsnittslønn for Ifi.

2.6.4 Et eksempel

Gitt tabellen `ansatt(anr, navn, lonn, avd)`, finn antall ansatte ved Ifi som tjener mer enn det dobbelte av snittlønna i administasjonen.

```
1 select count(*)
2 from ansatt
3 where avd = 'ifi' and
4   lonn > ( select 2 * avg(lonn)
5     from ansatt
6     where avd = 'adm' );
```

code/eks1.sql

Merk: en `select` inne i en `where`-betingelse må være omsluttet av parenteser.

2.6.5 group by

Finn antall ansatte i hver avdeling og snittlønn for disse:

- `ansatt(anr, navn, lonn, avd)`
- `avdeling(avdnr, avdnavn, leder)`
- `prosjektplan(pnr, anr, timer)`

```
1 select avdnavn, count(*), avg(lonn)
2 from ansatt, avdeling
3 where avd = avdnavn
4 group by avdnavn;
```

code/group.sql

Dette forutsetter at `avdnavn` er en kandidatnøkkel.

i SQL	betyr
exists R	at R har minst en forekomst
not exists R	at R ikke har noen forekomster
in R	$\in$ R
not in R	$\notin$ R
any R	en vilkårlig verdi i R
all R	alle verdier i R

Tabell 2: Relasjonssammenhenger.

2.7 Relasjonssammenhenger

2.7.1 any og all

Eksempel: `Ansatt(anr, navn, lonn, avd)`, `Avdeling(avdnr, avdNavn, leder)`, `Prosjekt(pnr, anr, timer)`

```
1 select  count(*)
2 from  Ansatt
3 where avd = 'ifi' and
4   lonn > all (select  lonn
5         from  Ansatt
6         where avd = 'kjemi');
```

code/anyall.sql

Oppgave: finn antall ansatte ved Ifi som tjener mer enn samtlige på kjemi.

2.7.2 in og not in

Samme tabellene som over. Oppgave: finn navn på ansatte som ikke har ført noen prosjekttimer.

```
1 select  navn
2 from  Ansatt
3 where anr not in (select  anr
4           from  Prosjekt);
```

code/innot.sql

2.8 view

Eksempel på view etter tabellen `Prosjekt(pnr, anr, timer)`.

```
1 create view Innsats as
2   select anr, sum(timer) as timer
3   from Prosjekt
4   group by anr;
5
```

```sql
create view Bonus (anr, bonusbelop) as
  (select anr, 3000
  from Innsats
  where timer >= 50)
union
  (select anr, 1500
  from Innsats
  where timer >= 15 and timer < 50);
```

code/view.sql

Et view er en tenkt relasjon som vi bruker som mellomresultat i kompliserte SQL-beregninger.

2.9 Hengetupler

Når vi joiner to tabeller, kaller vi et tuppel som ikke har noen match i den andre relasjonen, et *hengetuppel*. Hengetupler blir ikke med i resultatet av en vanlig join, også kalt en **inner join**. Hvis vi ønsker å gjøre en join hvor vi beholder hengetuplene fra en eller begge tabellene, bruker vi en **outer join**.

2.10 natural join

```sql
select f.title
from film f
natural join filparticipation fp
natural join person p
where p.firstname = 'Francis Ford' and p.lastname = '
    Coppola' and fp.parttype = 'director';
```

code/naturaljoin.sql

Finner alle filmer i en filmdatabase som Francis Coppola har vært director for. Natural join merger tabellene film, filmparticipation og person, slik at vi unngår doble forekomster av tupler.