UNIVERSITETET
I OSLO

INF1300
Introduksjon til databaser

Dagens tema:

- Informasjonsbærende referansemåter
- Resten av realiseringsalgoritmen
- Sterk realisering
- Realisering versus modellering

Representasjon av begreper

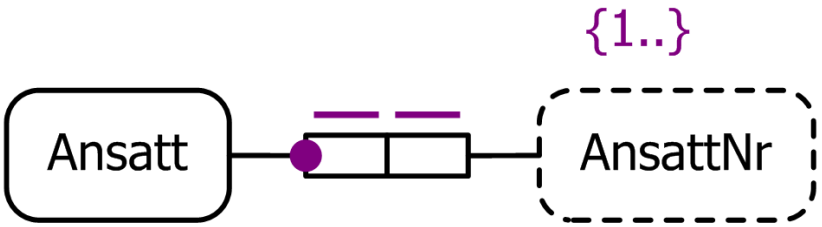
- Alle begreper må kunne representeres
  - Begrepsforekomster kan ikke lagres (de tilhører den virkelige verden, i form av fysiske eller abstrakte objekter); det vi lagrer, er *verdier som representerer forekomstene*
- Skal vi kunne realisere modellen som en database, må vi representere alle begrepene entydig
- Valg av representasjon - **referansemåte** - baserer seg til syvende og sist på et utvalg av de perfekte broene som finnes i modellen

Referansemåte – ønskelige egenskaper

- Helst **uforanderlig**, dvs. verdien som representerer en begrepsforekomst, endrer seg aldri
  - Referansemåten til et begrep blir til en primærnøkkel
  - Hvis en primærnøkkelverdi må endres, må også verdier i eventuelle fremmednøkler endres - og kanskje også verdier i andre, utenforliggende systemer. (Man har ikke nødvendigvis full oversikt lokalt over hvordan primærnøkler benyttes.)
- Støtte utveksling av informasjon mellom systemer
  - Bruk standardiserte referansemåter hvis de finnes (tid i UTC, vekt i kg, temperatur i °C, ...)

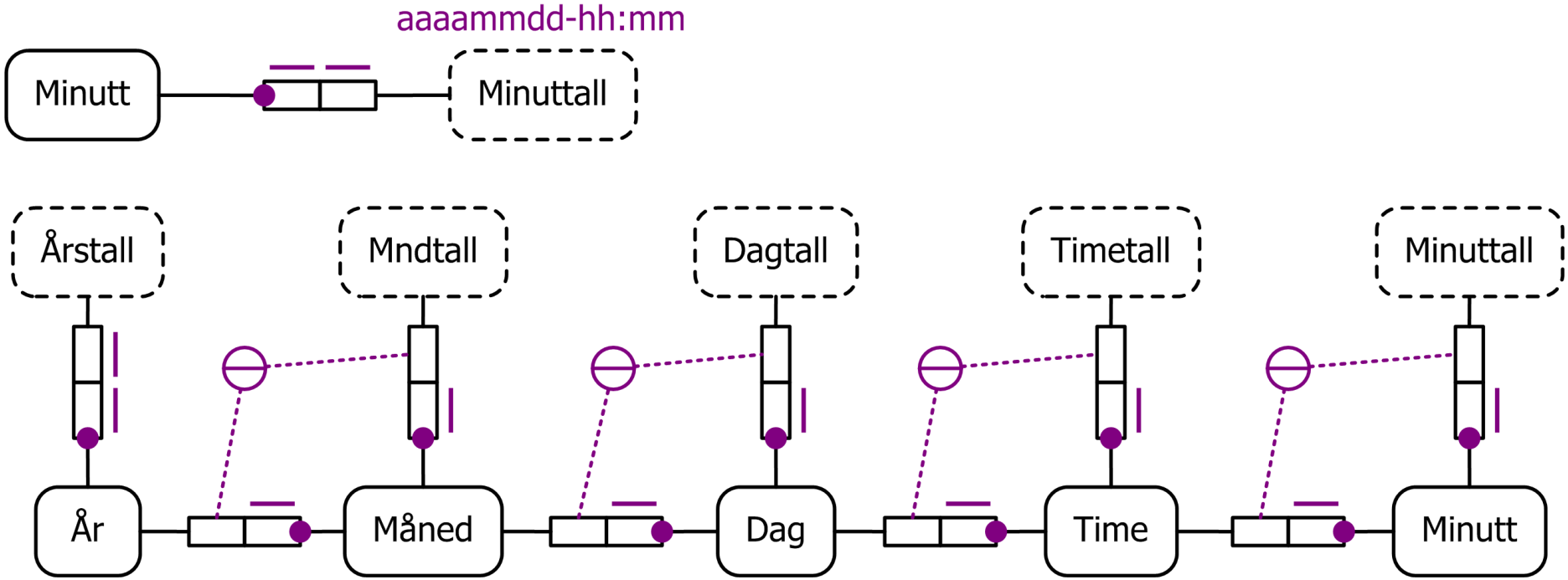
Ikke-informasjonsbærende referansemåter

- **Ikke-informasjonsbærende referansemåte**:
  - Referansemåten til begrepet peker ut én identifikator
  - Det finnes ingen innkodet informasjon i identifikatoren
- Det er enklest å opprettholde målet om uforanderlige referansemåter hvis det ikke ligger informasjon innkodet i referansemåten



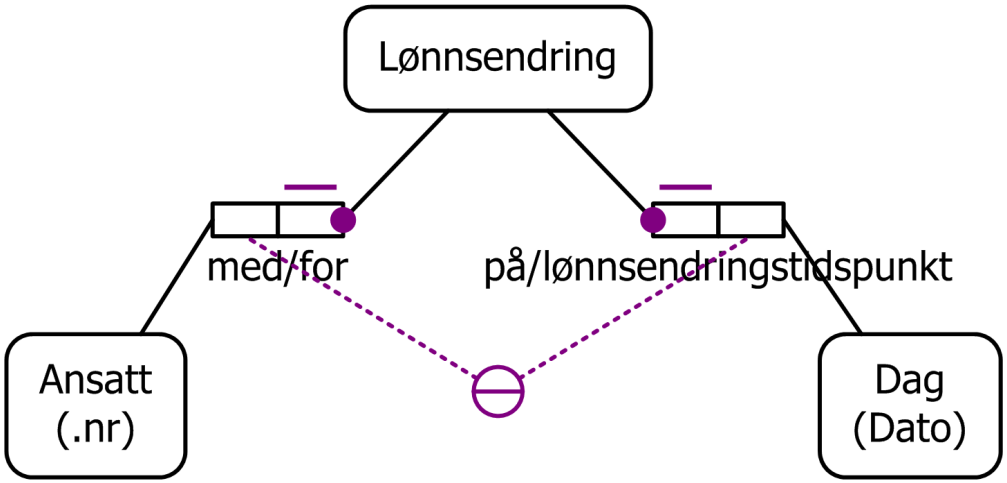
Delvis informasjonsbærende referansemåter

- **Delvis informasjonsbærende referansemåte**:
  - Deler av referansemåten til begrepet identifiserer en forekomst av et annet begrep
- Dette kan, men behøver ikke, være synlig i modellen



Totalt informasjonsbærende referansemåter

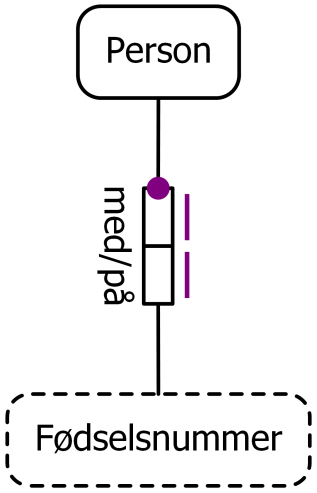
- **Totalt informasjonsbærende referansemåte**:
  - Referansemåten til begrepet baserer seg på en samling elementer der hvert element identifiserer en forekomst av et annet begrep



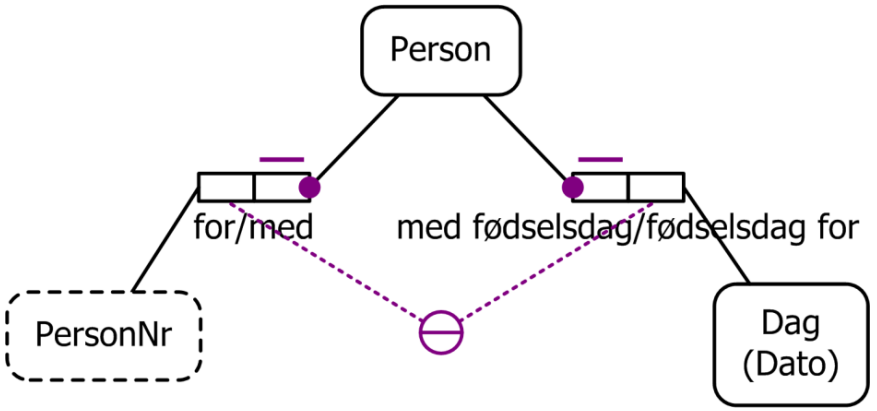
Synliggjøring eller ikke av informasjonsbærende referansemåte i modellen?

Hvis det er en mulighet for at brukeren etterspør denne informasjonen, bør den vises i modellen

Ikke synlig:



Synlig:



Realiseringsalgoritmen

Forberedelser:

1. Alle lange entydigheter gjøres om til nye begreper (med ekstern entydighet)
2. Alle begreper må kunne representeres
   (Forenklet: det må være nok perfekte broer i ORM-diagrammet)
3. Alle broer må ha en entydig begrepsrolle

Realiseringsalgoritmen:

1. For hvert begrep, lag en relasjon
2. For hvert begrep, velg en representasjon (fastsett referansemåte)
3. Behandle resten av broene
4. Behandle resten av faktatypene
5. Overfør resten av skrankene
6. Bestem hvilke undertrykkbare relasjoner som skal fjernes

Fra skranker til integritetsregler

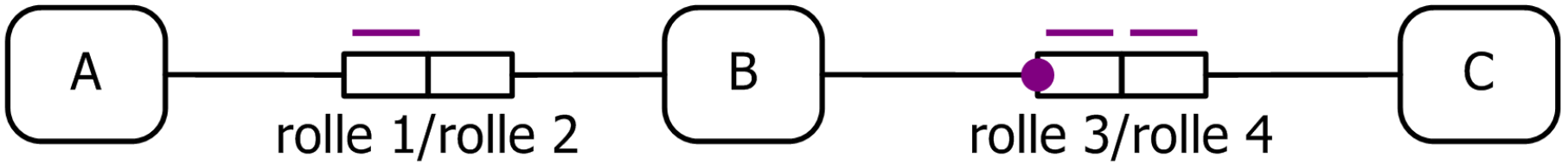
- Ved realisering fra en ORM-modell til et relasjonsskjema blir skranker i modellen omformet til integritetsregler i skjemaet
- De aller fleste skrankene overføres direkte til skjemaet
- Noen få ganger kan det være fornuftig å la være å håndheve en skranke i databasen, men dette er unntak
- **Vi skal *aldri* ha en integritetsregel i skjemaet som *ikke* kommer fra en skranke i modellen**

  Skjemaet får aldri være strengere enn modellen; modellen er systemets kontrakt med omverdenen

Realiseringsalgoritmen

- Repetisjon (og litt utdypning) av noen av de delene av algoritmen som allerede er forelest
- Gjennomgåelse av (det meste av) resten av algoritmen

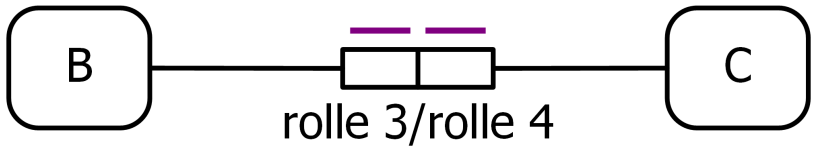
Gruppererroller og referanseroller



- I realiseringen vil faktatypen fra A til B bli til en fremmednøkkel fra A-relasjonen til B-relasjonen
- En-til-en-faktatypen mellom B og C blir til en entydig fremmednøkkel fra B-relasjonen til C-relasjonen. (Fremmednøkkelen plasseres i B fordi B-rollen er påkrevd.)
- Rollen til det begrepet som får fremmednøkkelen, kalles **gruppererrollen** i faktatypen (rolle 1 og 3)
- Rollen som blir til en fremmednøkkel, kalles **referanserollen** i faktatypen (rolle 2 og 4)

Håndtering av en-til-en-faktatyper uten påkrevde roller

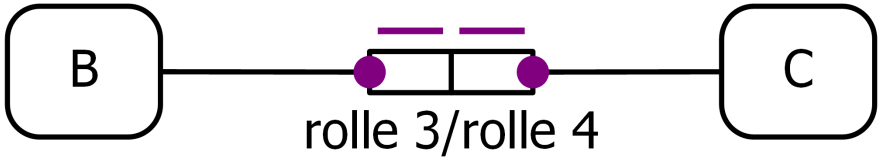
Her har vi et valg:



- Vi kan velge å peke ut en av rollene som gruppererrolle, dvs. hvilken relasjon som skal få fremmednøkkelen
- Vi kan velge å danne et begrep av faktatypen

  Resultatet blir i såfall en relasjon som består av to kandidatnøkler som er fremmednøkler til de to relasjonene som stammer fra begrepene som inngår i faktatypen. (En av kandidatnøklene velges som primærnøkkel.)

Håndtering av en-til-en-faktatyper med to påkrevde roller

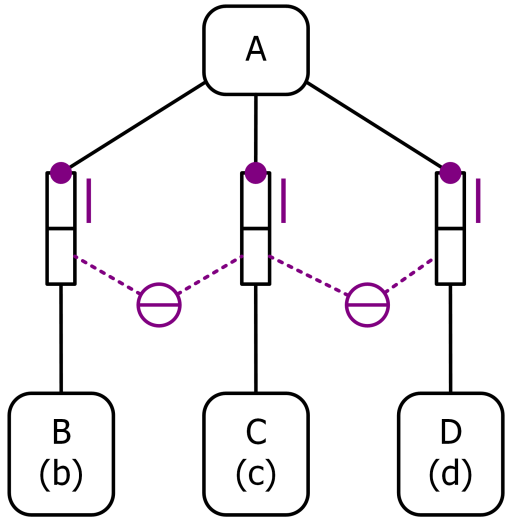


- Slike faktatyper forekommer sjelden

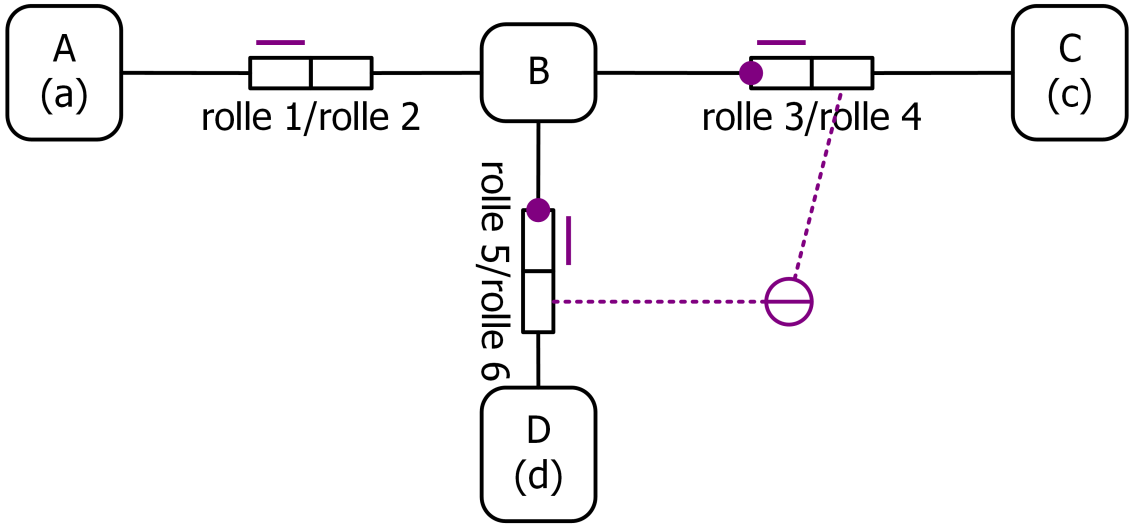
  Ett eksempel er en faktatype som kobler sammen hovedsteder med sine land
- Vi må da velge en av rollene som gruppererrolle
- Dersom ett av de involverte begrepene bruker denne faktatypen som sin referansemåte, har vi gjort en analysefeil. De to begrepene skal da slås sammen til ett begrep, og faktatypen skal fjernes fra ORM-modellen

Håndtering av eksterne entydigheter

- Eksterne entydigheter blir til kandidatnøkler
- En ekstern entydighet kan derfor brukes som referansemåte for et begrep

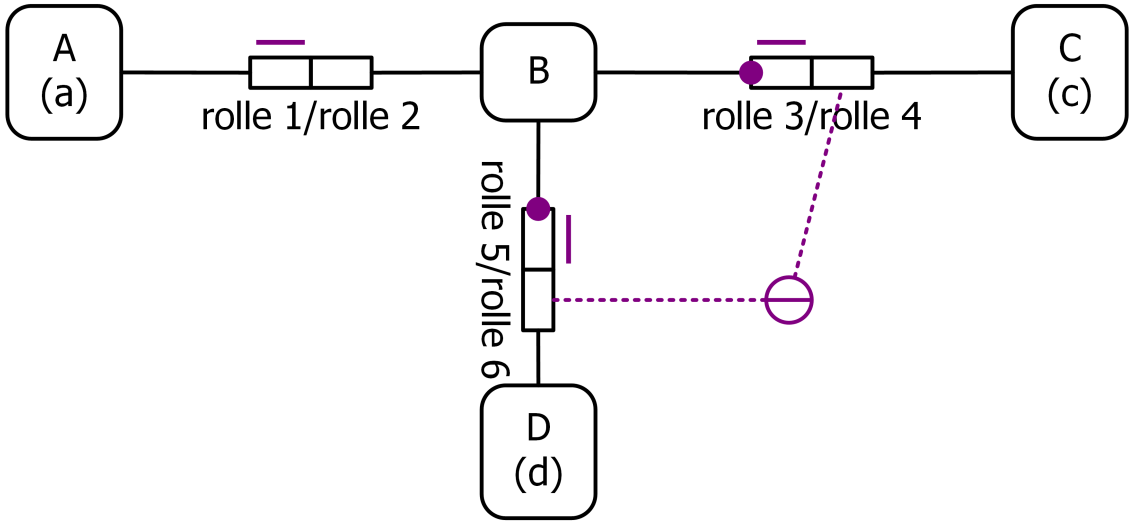


Navn på attributter – 1



- I eksempelet over vil A-relasjonen få en fremmednøkkel til B-relasjonen bestående av to attributter: ett C-attributt og ett D-attributt

Navn på attributter – 2

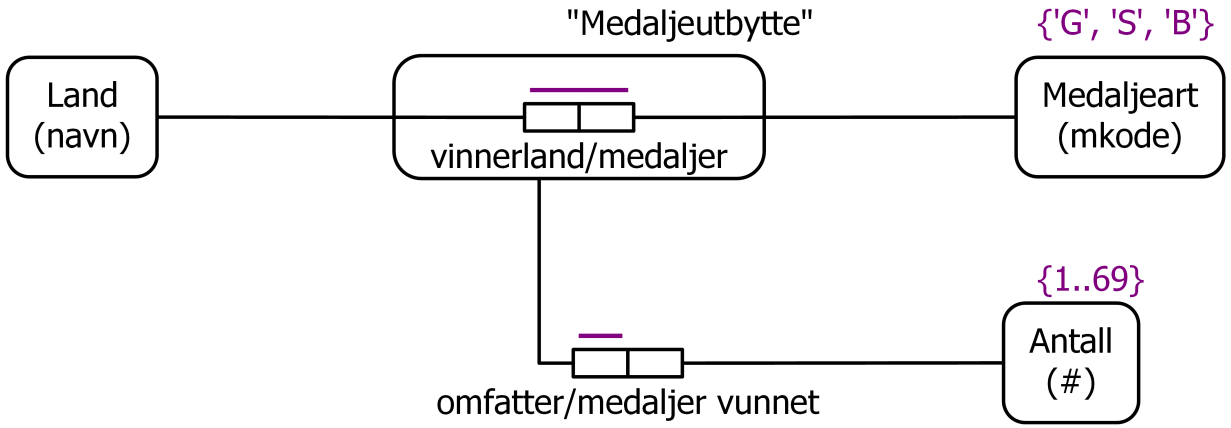


- Dere kan fritt velge navnene på de to attributtene i fremmednøkkelen i A. Det kan dere forøvrig gjøre for alle andre attributter også (men det bør være rimelig opplagt utifra attributtnavnene hva attributtene svarer til i ORM-modellen)
- En algoritmisk måte å bestemme navn på, er å kombinere referansemåte med rollenavn: c_rolle2 og d_rolle2 hvor c og d er referansemåtene til henholdsvis C og D

Håndtering av underbegreper

- Et underbegrep arver alltid referansemåten til sitt superbegrep
- Anta:
  - Alle begrepshierarkier har underbegrepsforklaringer
  - Alle underbegrep har minst én gruppererrolle
- Til hvert underbegrep opprettes en relasjon med samme primærnøkkel som superbegrepet, og med fremmednøkkel fra underbegrepets primærnøkkel til superbegrepets.
- To alternativer:
  1. Attributter fordeles mellom superbegrepets og underbegrepets relasjoner - *må joine for å finne alle data*
  2. Alle attributtene fra superbegrepets relasjon (unntatt den eller de faktatypene som inngår i underbegrepsforklaringen) gjentas i underbegrepets relasjon - *må håndtere dobbeltlagring*

Håndtering av verdiskranker



- En verdiskranke begrenser domenet for et attributt
- Verdiskranker kan i SQL uttrykkes ved hjelp av CHECK()
- Eksempelet over gir følgende definisjon av Medaljeutbytte:

```
CREATE TABLE Medaljeutbytte (
  land              CHAR(20) REFERENCES Land(navn),
  mkode             CHAR(1) CHECK( mkode IN ('G','S','B') ),
  antall            INTEGER CHECK( 0 < antall AND antall < 70 ),
  PRIMARY KEY(land, mkode)
);
```

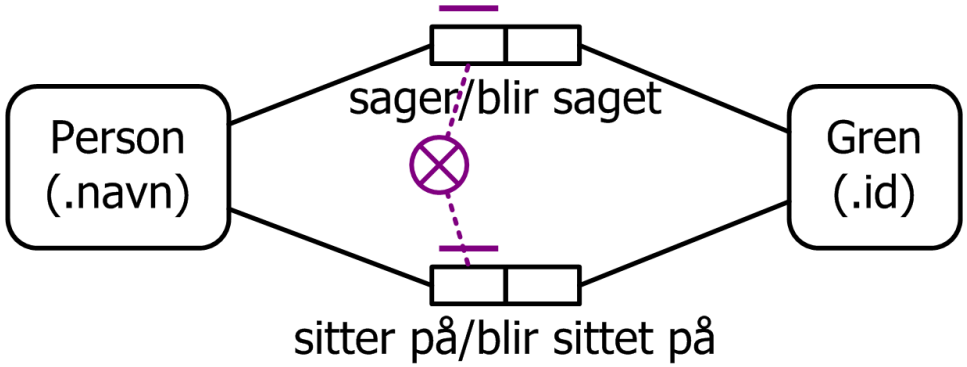
Forekomstrestriksjoner

- En integritetsregel som kan håndheves lokalt i en enkelt forekomst i en tabell, kalles en **forekomstrestriksjon**

  På engelsk kalles den en «**intra-record constraint**»
- En integritetsregel som ikke er en forekomstrestriksjon, har ikke noen egen betegnelse på norsk

  På engelsk kalles den en «**inter-record constraint**»
- Forekomstrestriksjoner er billige å håndheve fordi vi ikke trenger å lese noen andre forekomster enn den vi holder på med
- Forekomstrestriksjoner kan bare brytes ved innlegging og endring, aldri ved sletting

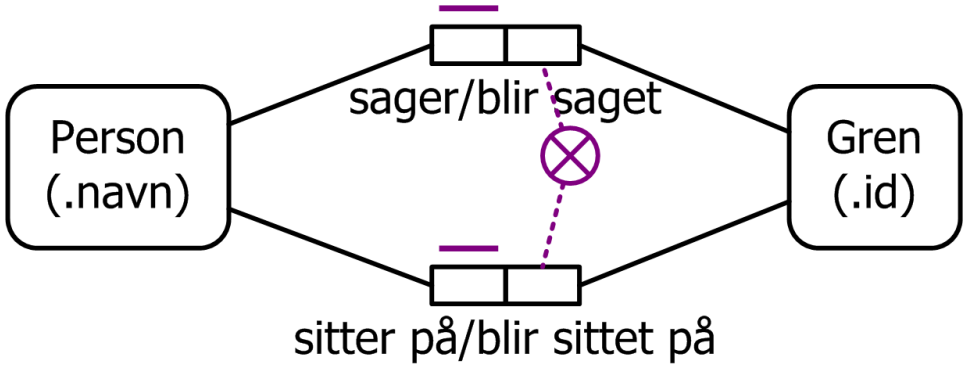
Ulikhet i gruppereroller



Person(personnavn, gren_blir_saget, gren_blir_sittet_på)

- Her sammenligner vi to gruppererroller for Person, og disse skal ikke ha noen felles forekomst
- Det betyr at ingen forekomst av Person skal finnes både i rollen sitter_på og i rollen sager
- Når vi legger inn en ny forekomst av Person, må vi altså kontrollere at minst ett av attributtene gren_blir_saget og gren_blir_sittet_på er NULL

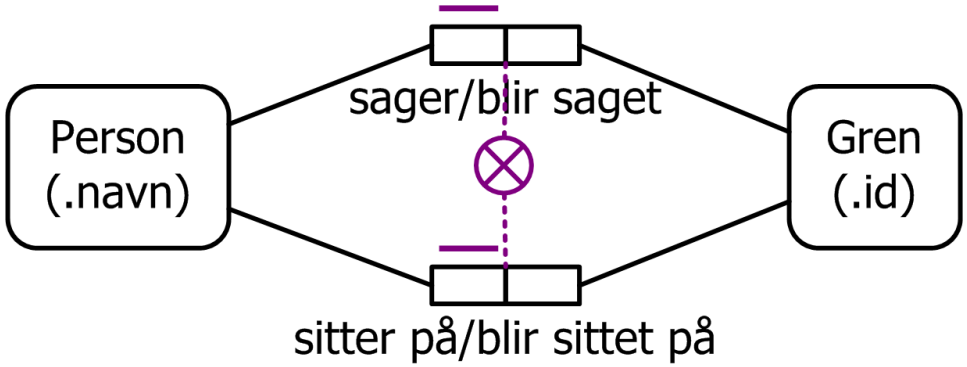
Ulikhet i referanseroller



Person(personnavn, gren_blir_saget, gren_blir_sittet_på)

- Her sammenligner vi to referanseroller til Gren, og disse skal ikke ha noen felles forekomst
- Når vi legger inn en ny forekomst av Person, må vi kontrollere at
  1) verdien i gren_blir_saget ikke finnes som verdi i gren_blir_sittet_på i noen forekomster
  2) verdien i gren_blir_sittet_på ikke finnes som verdi i gren_blir_saget i noen forekomster

Dobbeltrolleulikhet



Person(personnavn, gren_blir_saget, gren_blir_sittet_på)

- I dette tilfellet har vi en dobbeltrolleskranke hvor de to gruppererrollene tilhører samme begrep
- Skranken sier at ingen forekomstpar av Person og Gren skal ha samme verdi i rolleparene (sager, blir saget) og (sitter på, blir sittet på)
- Når vi legger inn en ny forekomst av Person, må vi altså kontrollere at attributtene gren_blir_saget og gren_blir_sittet_på har forskjellig verdi eller at minst ett av dem er NULL

Ulikhetsskranker og kombinerte påkrevde roller

- Enkeltrolleulikheter som bare går mellom gruppererroller, blir til forekomstrestriksjoner som bare ser på NULL.
- Dette gjelder også kombinerte påkrevde roller mellom gruppererroller
- Dobbeltrolleulikheter hvor ett begrep har begge gruppererrollene, blir til forekomstrestriksjoner hvor verdiene i attributtene sammenlignes
- Ingen andre ulikhetsskranker eller kombinerte påkrevde roller blir til forekomstrestriksjoner
- En generell kombinert påkrevd rolle som involverer flere referanseroller, er svært dyr å håndheve, og den er derfor kandidat til en skranke vi velger å neglisjere

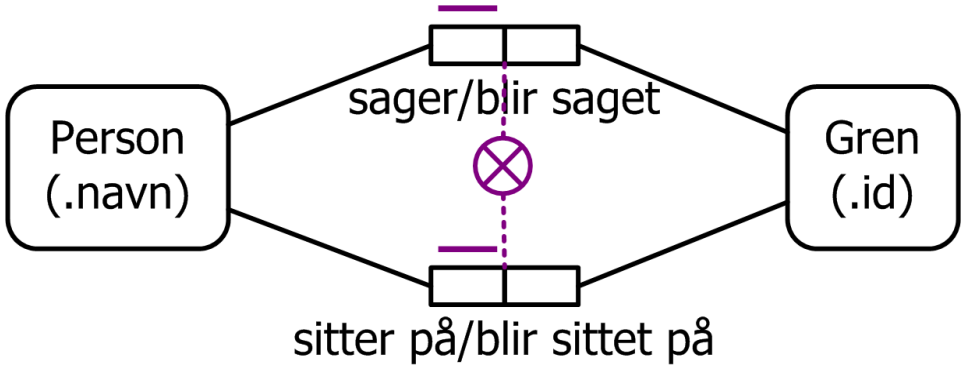
Håndtering av andre skranker I

- Håndtering av delmengdeskranker og likhetsskranker blir helt tilsvarende håndteringen av ulikhetsskrankene
- Ekvivalente stier (EOP) blir en slags likhetsskranke
  - Noen ganger havner de to attributtene som skal ha lik verdi i samme tabell; da er det enklest å slå sammen de to attributtene til ett
  - I mer kompliserte tilfeller havner de i forskjellige tabeller; da må alle tabeller langs de to stiene joines for deretter å sjekke at de to attributtene har lik verdi

Håndtering av andre skranker II

- Fremmednøkler er spesialtilfeller av delmengdeskranker
- At A har en fremmednøkkel til B, håndheves som to integritetsregler:
  - Ved INSERT på A må vi sjekke at fremmednøkkelen har en lovlig verdi (peker på en forekomst av B)
  - Ved DELETE av en B må vi sjekke at ingen A har en fremmednøkkel til denne forekomsten av B

Forekomstrestriksjoner i SQL



Person(personnavn, gren_blir_saget, gren_blir_sittet_på)

- Alle forekomstrestriksjoner kan i SQL uttrykkes ved hjelp av CHECK()
- Eksempel: Når vi legger inn en ny forekomst i Person, må minst ett av attributtene gren_blir_saget og gren_blir_sittet_på være NULL, eller de må ha forskjellig verdi

  I SQL uttrykkes dette slik (i table Person):

```
CHECK( gren_blir_saget IS NULL OR
       gren_blir_sittet_på IS NULL OR
       gren_blir_saget <> gren_blir_sittet_på )
```

Andre integritetsregler i SQL

- Vi har tidligere sett hvordan vi definerer
  - Primærnøkler
  - Andre entydighetsskranker (dvs. kandidatnøkler)
  - Fremmednøkler
- Øvrige integritetsregler må vi programmere selv i form av databasefunksjoner som kalles fra triggere

Triggere og triggerprogrammer

- En **trigger** utløses av en hendelse som INSERT, DELETE eller UPDATE i databasen
- Triggere kan knyttes opp mot databasefunksjoner (triggerprogrammer)
- Når triggeren utløses, utføres den tilhørende databasefunksjonen
- Hvordan man programmerer triggere og databasefunksjoner, er svært DBMS-avhengig
- Triggerprogrammering er ikke pensum

Sterk realisering – 1

- Ved vanlig realisering vil ikke-påkrevde gruppererroller gi attributter hvor NULL er tillatt
- Det å **realisere sterkt** betyr at **nullverdier aldri er tillatt**
- For å få til dette, må relasjonene splittes:
  - Vi får én relasjon med de attributtene som kommer fra faktatyper med påkrevd gruppererrolle
  - Faktatyper hvor gruppererrollen ikke er påkrevd, blir til en egen relasjon bestående av primærnøkkelen og attributtet fra denne faktatypen
  - Ved likhetsskranke mellom gruppererroller kan attributtene fra disse faktatypene legges i samme relasjon

Sterk realisering – 2

- Sterk realisering gir samme resultat som det å opprette et underbegrep for hver ikke-påkrevde gruppererrolle
- Det gjør at alle gruppererroller blir påkrevde ved sterk realisering
- Resultatet blir en relasjonsdatabase hvor ingen attributter kan være NULL

Begreper som kan undertrykkes

- Et begrep som ikke spiller noen andre gruppererroller enn de som inngår i referansemåten, og som spiller minst én referanserolle, kalles et **undertrykkbart begrep**
- Tabeller som kommer fra slike begreper, kan fjernes (**undertrykkes**) fra relasjonsdatabaseskjemaet

Realisering versus modellering

- Følgende er en del av **realiseringsprosessen** og *ikke* av modelleringsprosessen:
  1. Valg av referansemåter og hvilke datatyper som skal knyttes til identifikatorene
  2. Valg av gruppererroller
  3. Valg av om det skal brukes sterk realisering
  4. Valg av hvordan underbegreper skal realiseres
  5. Valg av hvilke relasjoner som skal undertrykkes, og hvilke som skal beholdes
- Realiseringsprosessen er automatiserbar og kan gjøres av en datamaskin
  - Man kan lage generelle regler for hvordan 1-5 skal håndteres og få en helautomatisert prosess, *men databaseskjemaet blir bedre hvis det gjøres bevisste valg der hvor det er mer enn én mulighet*