UNIVERSITETET I OSLO

IN2090
Introduksjon til databaser

Dagens tema:

- Data, databaser og databasehåndteringssystemer
- Hva er data? Hva er informasjon?
- Fra idé til informasjonssystem
- Litt om modellering:
  - Begreper og representasjon av dem
  - Elementære setninger

Neste forelesning (onsdag 29.8.):

- Mer om modellering

Data, databaser og databasehåndteringssystemer

Transiente og persistente data

- Når vi programmerer, legger vi dataene våre i programvariable (eller bare «variable»).
- Når programmet avsluttes, slettes programvariablene og dermed også dataene i dem.
- Slike data kalles **transiente**.
- En enkel måte å ta vare på data, er å skrive dem til en fil. Det vil vanligvis si at dataene bak kulissene lagres på en magnetisk harddisk (HDD) eller en solid state drive (SSD).
- Data som overlever mellom to programkjøringer, kalles **persistente**.

Problemer med filer

- I utgangspunktet er en fil ment å skulle brukes av ett program om gangen. Vi risikerer at hvis det er mulig for flere programmer å åpne og bruke filen samtidig, så kan endringer gjort av ett program bli ødelagt av andre, samtidige programmer, og på en måte som gjør at innholdet av filen ikke lenger gir mening.
- Det er heller ikke generelt noen garanti for at innholdet i filene ikke kan gå tapt hvis noe galt skjer med systemet (f.eks. diskkræsj).
- Filsystemer har i utgangspunktet ingen støtte for effektivt å gjenfinne dataelementer som ligger i en fil, hvis vi ikke vet nøyaktig hvor i filen dataene ligger.
- Store programmer trenger ofte mange typer data, noe som gjerne resulterer i mange filer. Programmet må selv holde oversikten over alle filene og blir derfor desto mer komplisert.

Databaser og DBMS

- En **database** er en samling persistente data som fysisk bare kan aksesseres fra ett spesialprogram, kalt et databasehåndteringssystem, forkortet **DBMS** (DataBase Management System).
- Vilkårlig mange programmer kan samtidig være koblet opp mot DBMSet og via det få lest, skrevet og endret data i databasen. DBMSet tar ansvar for å
  - lagre store mengder data over lang tid
  - persistere dataene, også i nærvær av systemfeil mm.
  - kontrollere dataaksess slik at mange programmer kan bruke og modifisere dataene samtidig uten å ødelegge for hverandre
  - effektivt gjenfinne data på vegne av programmene

Transaksjoner

- En henvendelse fra et program til DBMSet kalles en **transaksjon**
- En transaksjon som bare leser data, kalles en lesetransaksjon eller en **spørring** (**query**, uttales: kwi:ri)
- En transaksjon som legger til nye data eller forandrer eksisterende data, kalles en skrivetransaksjon

Litt historie

- Det første DBMSet ble utformet av Bachman og Williams i 1964
  - DBMSet ble solgt under navnet IDMS
  - IDMS var en *nettverksdatabase*, og var designet for bruk fra et programmeringsspråk (*vertsspråk*)
- I 1968 kom IBM med IMS, som var en *hierarkisk database* (en forenkling av nettverksdatabaser)
  - IMS ble nær enerådende for administrativ databehandling. Fortsatt er det svært mange store firmaer som har legacysystemer som bruker IMS

Litt mer historie

- I 1970 presenterte E.F.Codd sin relasjonsmodell
- Dette var en teoretisk beskrivelse av en ny type databaser kalt *relasjonsdatabaser*
- Relasjonsdatabaser er enkle å beskrive og bruke, men vanskelige å lage et DBMS for
- Først i 1977 klarte Oracle å lage et DBMS som fortjener betegnelsen relasjonell
- Relasjonsdatabaser har vært svært mye brukt i mange tiår nå, og de er hovedtemaet for dette kurset

Hva er data?
Hva er informasjon?

Data

- Fra programmeringsspråk er vi vant til at vi har forskjellige datatyper som
  - int (heltall)
  - double (desimaltall)
  - char (tegn)
- Heltallsvariable kan ha verdier som 17 og -1024
- Slike verdier kaller vi **data**
- En enkelt verdi heter egentlig et datum, men vi bruker som oftest ordet «data» i entall også

Informasjon

- **Informasjon** består av data pluss regler for hvordan data skal tolkes
- Hvis vi har en (desimaltalls-) variabel *vekt* med verdien 54,2, vil det være naturlig å anta at 54,2 er vekten av ett eller annet
- Men for at 54,2 skal kunne kalles informasjon, er det to spørsmål som må besvares:
  - Hvilken måleenhet er brukt for vekt?
  - Hva er det som veier 54,2 måleenheter?

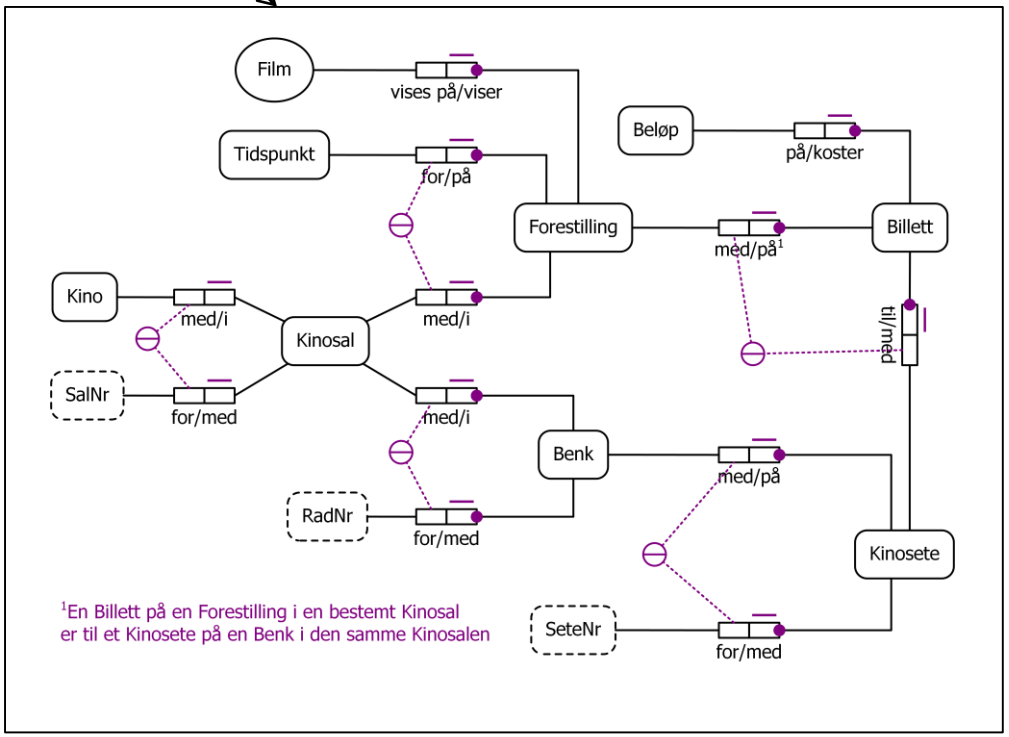
Fra idé
til informasjonssystem

Billettsystem for kino

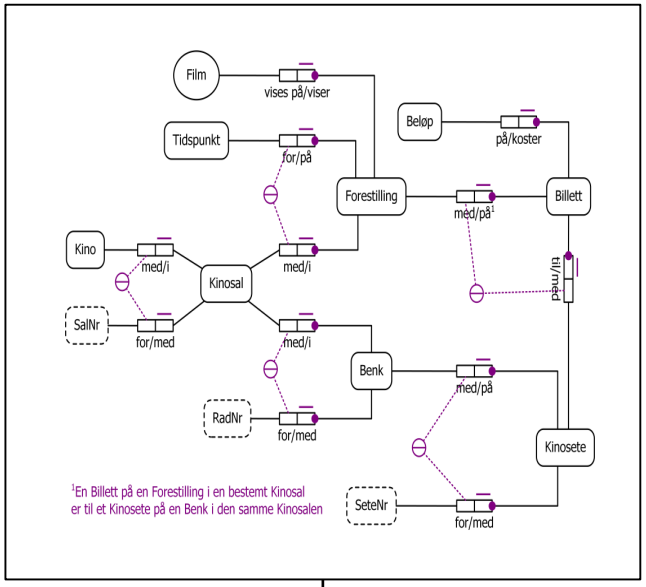
Interesse-område

- Kan det være mer enn én sal pr. kino?
- Kan en billett brukes til å se mer enn én forestilling?
- Er det alltid én billett pr. person, eller kan det være gruppebilletter også?
- Gjelder en billett alltid ett eller flere gitte seter, eller er det noen forestillinger der man kan sette seg hvor man vil?
- ...

Analyse og informasjons-modell



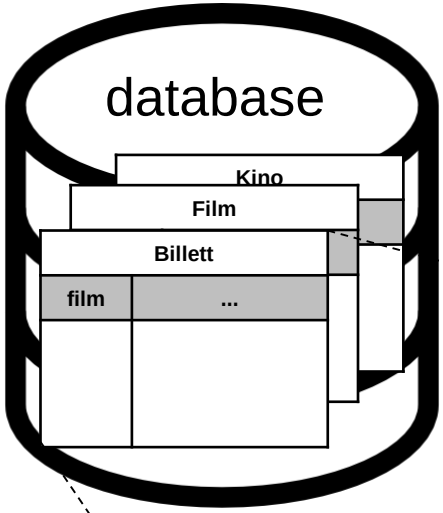
Realisering 1: Fra informasjons-modell til skjema

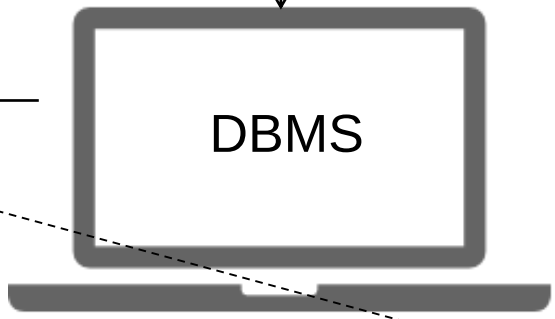


```
CREATE TABLE Billett (
  film VARCHAR(100) NOT NULL,
  tidspunkt TIMESTAMP,
  kino VARCHAR(30),
  salnr VARCHAR(10),
  radnr INT,
  setenr INT,
  pris INT NOT NULL,
  PRIMARY KEY (tidspunkt, kino, salnr, radnr, setenr)
);
CREATE TABLE ...
```

Realisering 2: Opprettelse av en database

```
CREATE TABLE Billett (
  film VARCHAR(100) NOT NULL,
  tidspunkt TIMESTAMP,
  kino VARCHAR(30),
  salnr VARCHAR(10),
  radnr INT,
  setenr INT,
  pris INT NOT NULL,
  PRIMARY KEY (tidspunkt, kino, salnr, radnr, setenr)
);
CREATE TABLE ...
```



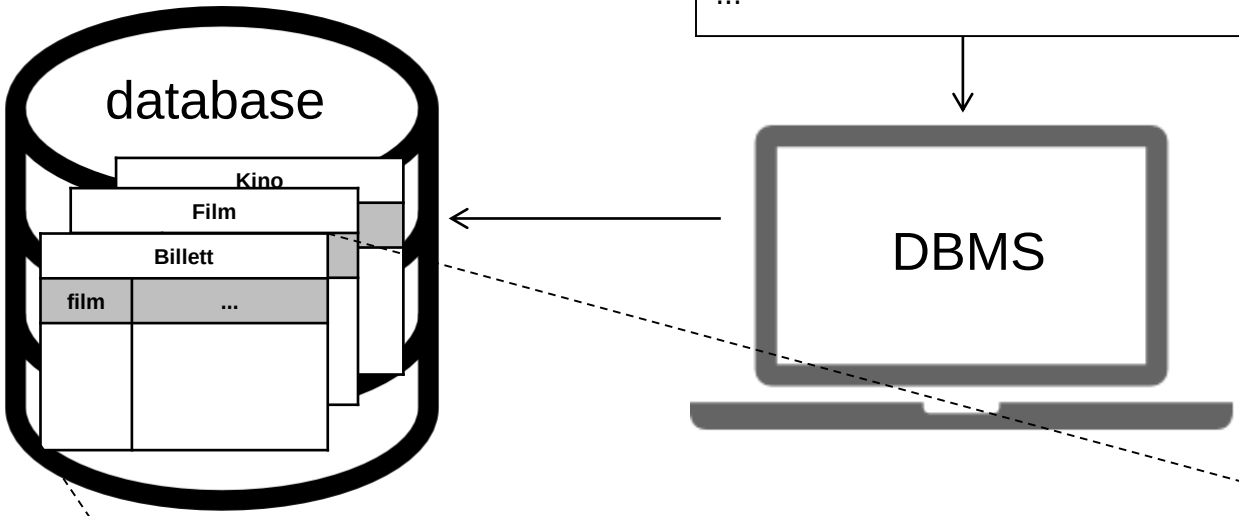


Billett

film	tidspunkt	kino	salnr	radnr	setenr	pris

Innlegging av data i databasen

```
INSERT INTO Billett
        VALUES ('七人の侍', 2017-...);
INSERT INTO Billett
        VALUES (...);
...
```



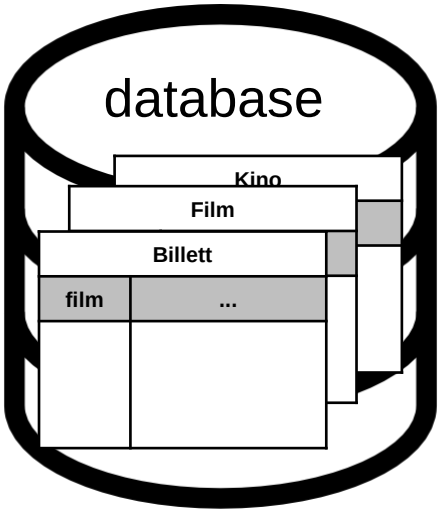
Billett

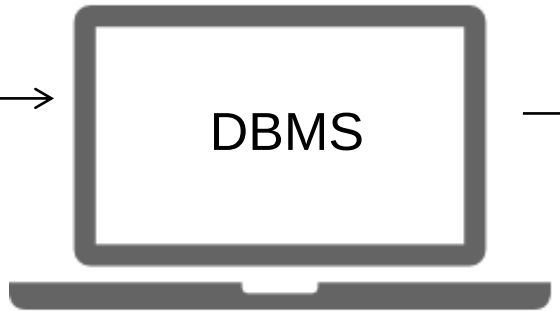
film	tidspunkt	kino	salnr	radnr	setenr	pris
七人の侍	2017-...	Saga	Sal 1	...	...	...
...						

Spørringer

Hvor mange billetter ble det i gjennomsnitt solgt til forestillinger som viste filmen 七人の侍 på Saga kino i tidsrommet 1.-15. august 2017?

```
SELECT COUNT(*)/SUM(antbill) AS gjsnitt
FROM (SELECT tidspunkt, salnr,
             COUNT(*) AS antbill
      FROM Billett
      WHERE kino = 'Saga' AND
            film = '七人の侍' AND
            tidspunkt >= 2017-08-01 AND
            tidspunkt <= 2017-08-15
      GROUP BY tidspunkt, salnr) AS f;
```

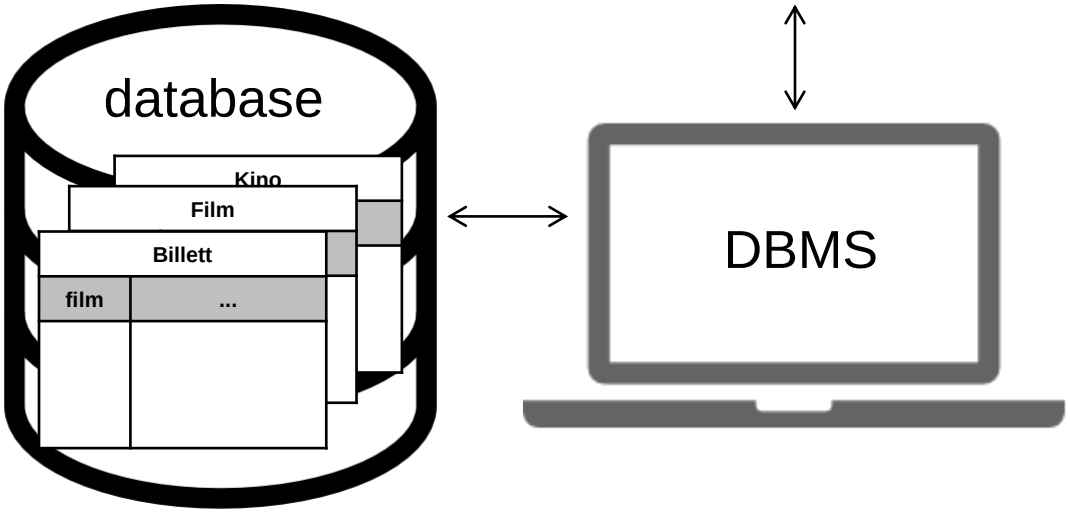





```
gjsnitt
--------
    73
(1 row)
```

Tilgang til databasen fra et applikasjons-program

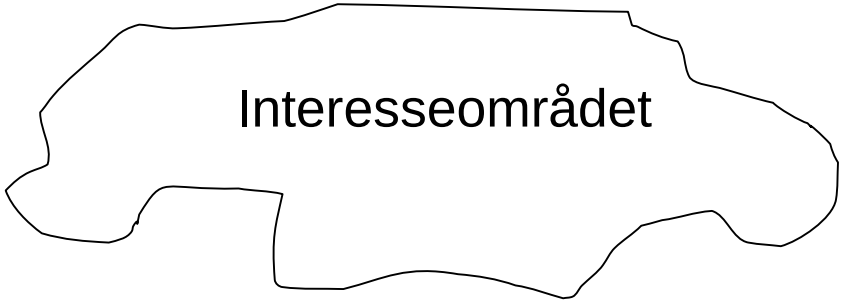
```
integer gjennomsnitt;
Connection myCon = DriverManager.getConnection(...);
Statement execStat = myCon.createStatement();
ResultSet resultat =
     execStat.executeQuery("SELECT COUNT(*)/SUM(antbill) AS gjsnitt FROM ...");
gjennomsnitt = resultat.getInt(1);
…
```



Interesseområdet (UoD = Universe of Discourse)

- Lovene som styrer virkeligheten, kaller vi **forretningsregler**
- Forretningsregler og naturlover har mange likhetstrekk
- Vi ser effekten av dem, men de kan være vanskelige å finne

Beskrivelse (deskripsjon) av virkeligheten/interesseområdet



Analyse:
Begrepsdannelse og idealisert representasjon

Beskrivelse av interesseområdet

Informasjonsmodeller

- En fullstendig beskrivelse av interesseområdet kalles en **informasjonsmodell**
- Informasjonsmodellen uttrykkes gjerne i et **modellspråk**
- Noen aktuelle modellspråk er
  - UML (Unified Modelling Languages)
  - ORM (Object-Role Modelling)
  - ER (Entity Relationship)
  - Logikk (utsagnslogikk, første ordens…)
  - Ontologispråk
- *Vår beskrivelse av UoD skal leses av en datamaskin, så den må være meget nøyaktig og detaljert*

Skranker

- Beskrivelsen av forretningsreglene kalles **skranker**
- *Statiske skranker* beskriver begrensninger på mulige tilstander i interesseområdet
- *Dynamiske skranker* beskriver begrensninger på mulige forandringer i interesseområdet
- Den ferdige ORM-modellen er en beskrivelse av de statiske skrankene i vårt UoD

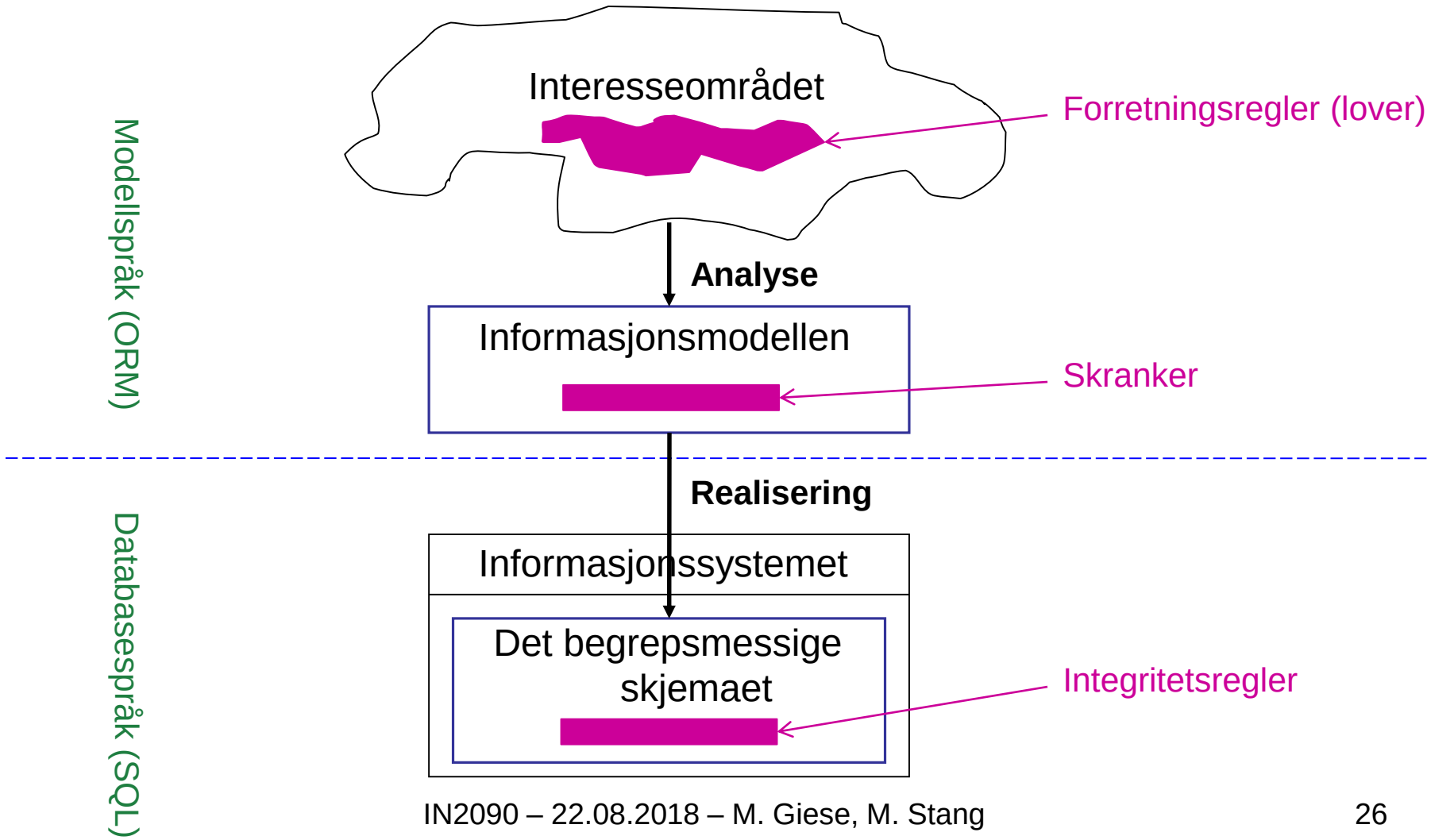
Det begrepsmessige skjema

- Informasjonsmodellen brukt som regelverk (*preskripsjon*) for hvordan informasjonssystemet skal oppføre seg, kalles **det begrepsmessige skjema** (eller bare **skjema**)
- Det begrepsmessige skjema uttrykkes i et språk som passer for den databaseteknologien vi skal bruke, f.eks.
  - SQL (Structured Query Language) for relasjonsdatabaser
  - ODL (Object Definition Language) for objektdatabaser

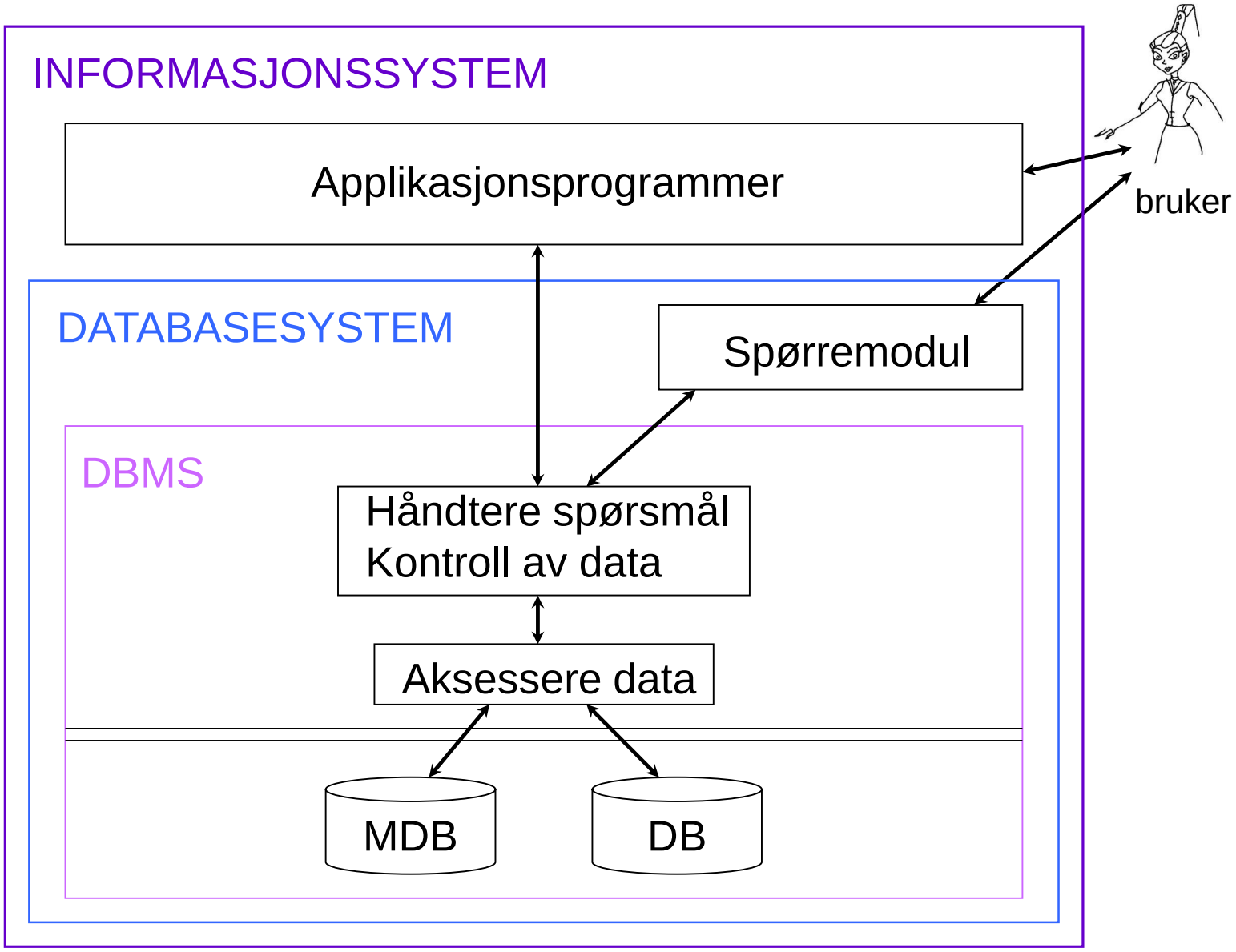
Integritetsregler

- I det begrepsmessige skjemaet kaller vi skrankene for **integritetsregler**
- Integritetsreglene bestemmer
  - hva som er lovlig å *lagre* i informasjonssystemet (lovlige tilstander i databasen)
  - hva som er lovlige *forandringer* (lovlige transisjoner/transaksjoner)
- Det er umulig for en bruker av informasjonssystemet å gjøre noe som strider mot integritetsreglene (det er DBMSets ansvar å passe på at integritetsreglene overholdes)

Informasjonssystemer



Informasjonssystemer vs. databaser



Litt om modellering

ORM – Object Role Modelling

- **Modellspråk**
- **Modelleringsmetode**

Tre viktige prinsipper:

1. **Ogdens trekant:** Sammenhengen mellom virkelighet og modell
2. **Naturlig språk:** Menneskespråk/naturlig språk er det mest fundamentale kommunikasjonsverktøyet vi har. Modellen må kunne uttrykkes i naturlig språk for å sikre at den kan forstås fullt ut av informerte brukere (de som kjenner virksomhets-området)
3. **100%-prinsippet:** Vi kan lage en nøyaktig nok modell av virkeligheten ved hjelp av naturlig språk

Begreper

- Vi setter navn på grupper av tilsvarende ting
- Vi skiller på den *fysiske manifestasjonen* av noe og *begrepet* vi bruker om det
- I denne salen er det mange personer – **begrepet** vi bruker er «person»
- To mer abstrakte eksempler er *lån* og *flyavgang*

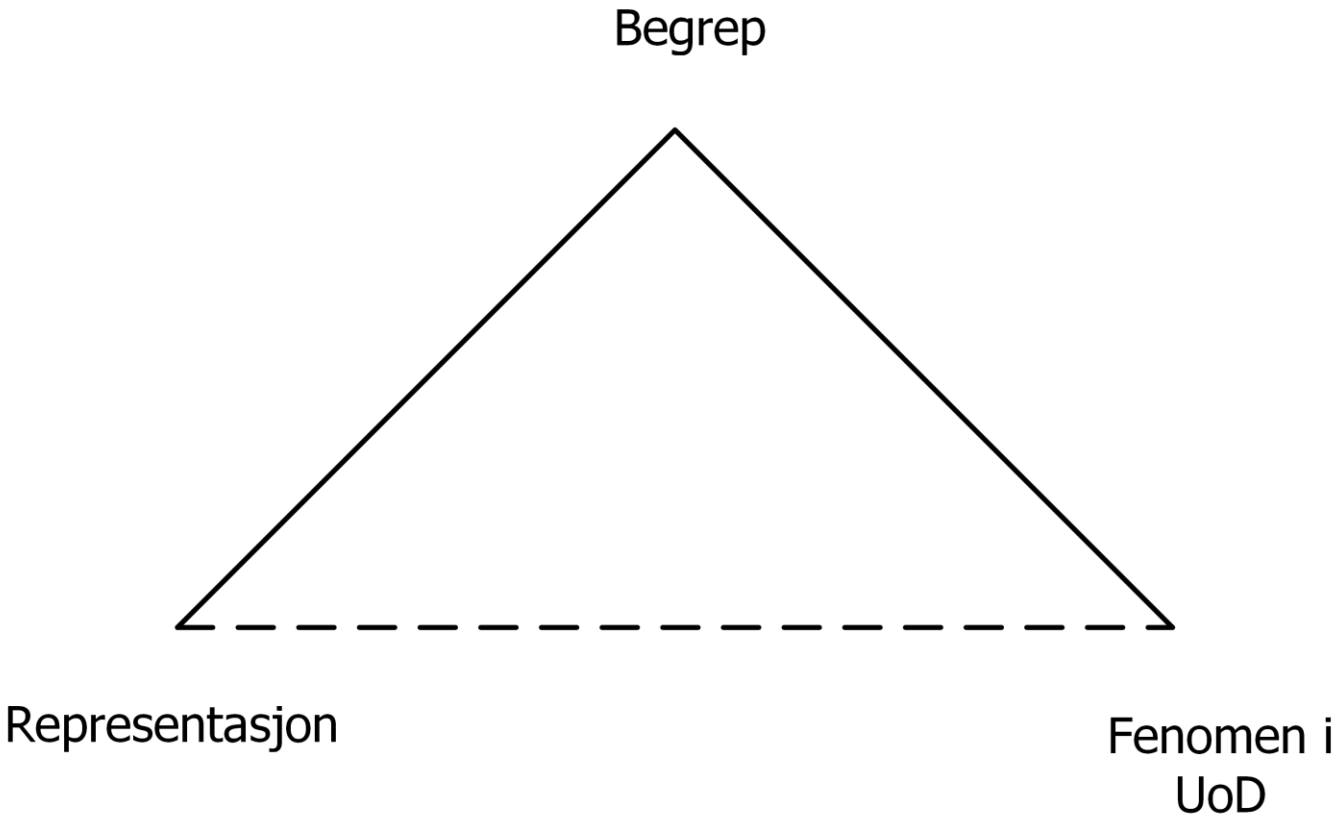
Representasjon

- Vi kan ikke lagre personer i databasen, så vi må finne en måte å lagre informasjon som identifiserer hver enkelt person
- En slik identifikasjonsmåte kalles en **representasjonstype** eller **verditype** for begrepet
- Når vi modellererer et *UoD*, må vi velge hvilke begreper modellen skal inneholde
- I tillegg må vi for hvert begrep bestemme oss for hvordan vi skal lagre informasjon om forekomster av dette begrepet

Ontologi

- Vitenskapen om sammenhengen mellom virkelighet (UoD), begreper og representasjon kalles **ontologi**
- Ontologi er et meget aktivt forskningsfelt
- Ett eksempel er oljeindustrien som prøver å bli enige om felles begreper og representasjoner for å beskrive alle installasjoner og alt vedlikehold i Nordsjøen
- Hovedpoenget med ontologi betegnes gjerne som Ogdens trekant

Ogdens trekant



Elementære setninger

- En setning som ikke kan deles opp uten å miste meningsinnhold, kalles **elementær**
- Eksempel:
  - Bjarte tar IN2090 og IN2010

  Denne setningen er *ikke* elementær fordi den kan erstattes av de to elementære setningene
  - Bjarte tar IN2090
  - Bjarte tar IN2010

Setninger og Ogdens trekant

- Se på setningen «Hanne tar IN1000»
- Det er mye informasjon som er underforstått:
  - Hanne er navn på en student
  - IN1000 er en emnekode
- Det vi mener er at

  studenten med navn Hanne

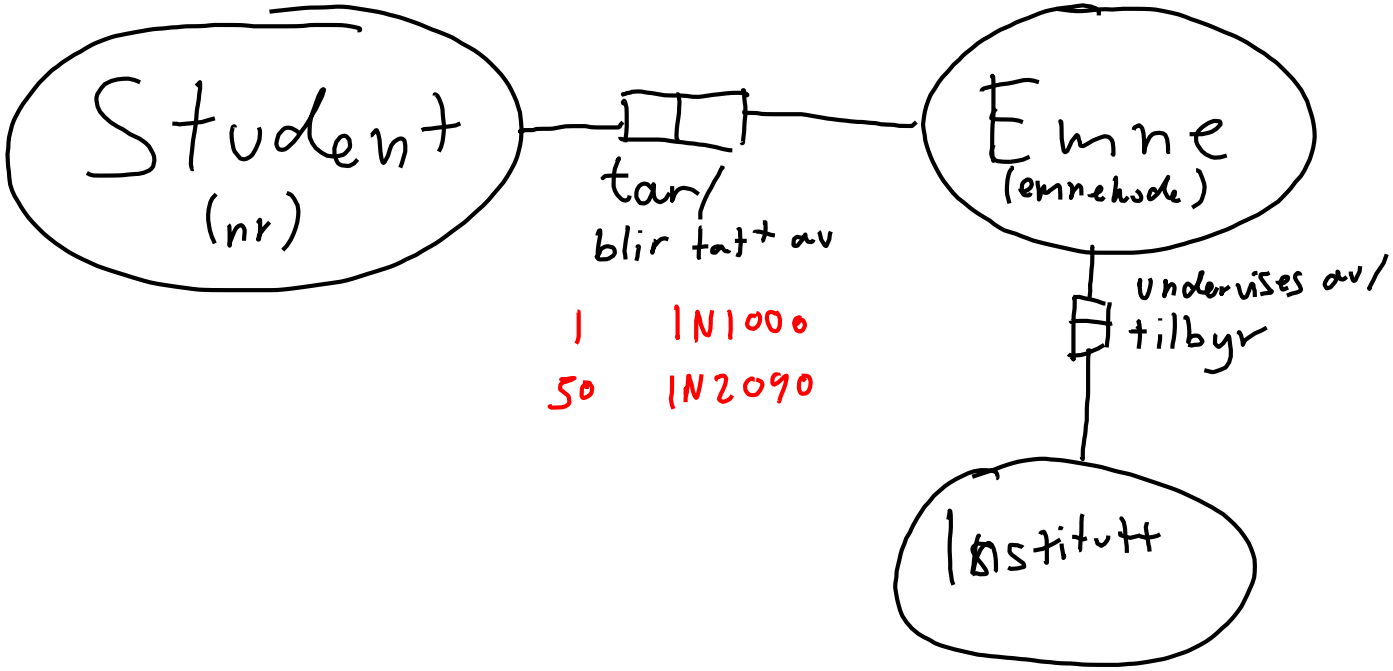
  tar emnet med emnekode IN1000

Setninger og faktatyper

- La oss se nærmere på setningen: «Studenten med navn Hanne tar emnet med emnekode IN1000»
- «student» og «emne» er **begreper**
- «navn» og «emnekode» er deres **representasjonstyper** (**verdityper**)
- «Hanne» og «IN1000» er **forekomster** (data)
- Utsagnet kaller vi *fakta* og det er dette vi modellerer – i ORM kaller vi det **faktatyper**

En faktatype i ORM

- «Student tar emne»



Setninger og faktatyper

- De to setningene under har samme meningsinnhold:
  - «Studenten med navn Hanne tar emnet med emnekode INF1000»
  - «Emnet med emnekode INF1000 har deltaker studenten med navn Hanne»
- Vi kan forme liknende **fakta** ved å bytte ut forekomstene:
  - «Studenten med navn Henrik tar emnet med emnekode INF1100» (eller: «Emnet med emnekode INF1100 har deltaker studenten med navn Henrik»)
- I ORM tegner vi denne **faktatypen** slik: