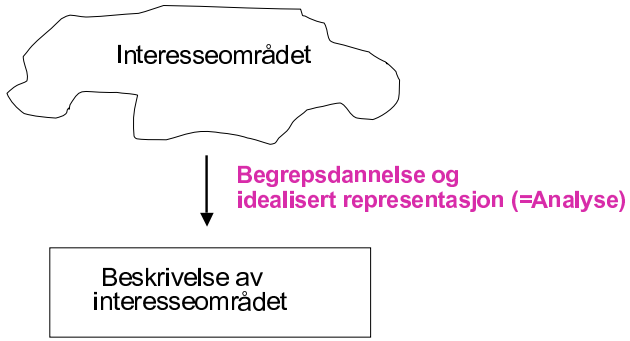
UNIVERSITETET I OSLO

Informasjonssystemer, DBMSer og databaser

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



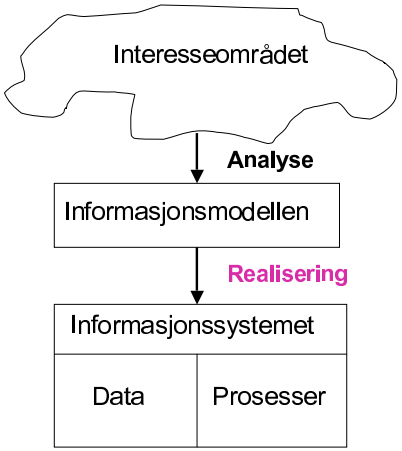
Informasjonsmodeller, 100%-prinsippet og skranker

- En fullstendig beskrivelse av interesseområdet kalles en **informasjonsmodell**
- *100%-prinsippet* sier at det er mulig å lage en (endelig) informasjonsmodell på norsk eller et annet naturlig språk
- Informasjonsmodellen uttrykkes gjerne i et *modellspråk* Aktuelle modellspråk er UMLs klassediagrammer, ER (Entity Relationship) eller ORM (Object-Role Modelling)
- Beskrivelsen av forretningsreglene kalles **skranker**
- *Statiske skranker* beskriver begrensninger på mulige tilstander i interesseområdet
- *Dynamiske skranker* beskriver begrensninger på mulige forandringer i interesseområdet

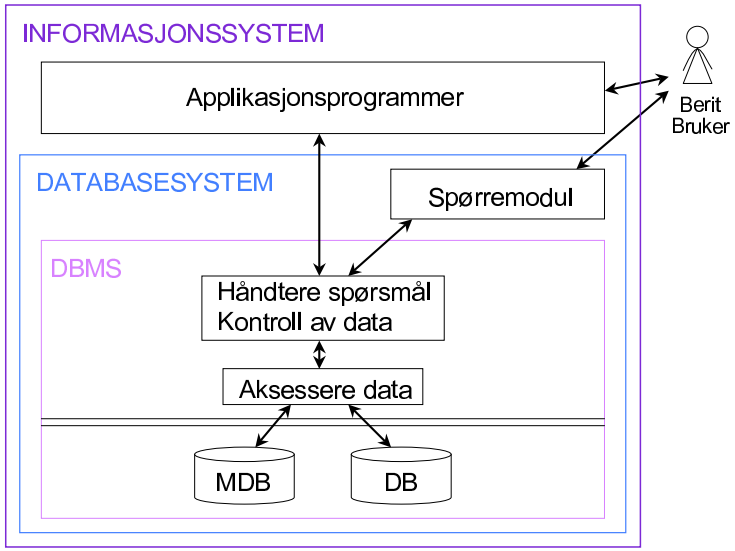
Det begrepsmessige skjema
Integritetsregler

- Informasjonsmodellen brukt som regelverk (*preskripsjon*) for hvordan informasjonssystemet skal oppføre seg, kalles **det begrepsmessige skjema**
- Det begrepsmessige skjema uttrykkes i et språk som passer for den databaseteknologien vi skal bruke, f.eks. SQL (Structured Query Language) for relasjonsdatabaser
- I det begrepsmessige skjemaet kaller vi skrankene for **integritetsregler**
- Integritetsreglene bestemmer hva som er lovlig å *lagre* i informasjonssystemet (lovlige tilstander) og hva som er lovlige *forandringer* (lovlige transisjoner)

Informasjonssystemer



Informasjonssystemer vs. databaser



DBMS –
Database Management System

- Spesialisert SW
- Karakteristika:
  - Persistens
  - Transaksjonshåndtering
    - **A**tomicity
    - **C**onsistency
    - **I**solation
    - **D**urability
  - Programmeringsgrensesnitt (API)

Litt databasehistorie

(Kursorisk pensum)

Begynnelsen

- Bachman & Williams:
  A General Purpose Programming System for Random Access Memories [1964] beskrev IDS (Integrated Data Store)
- IDS var det første kommersielle DBMS (utviklet av General Electric, Bachman var prosjektleder)
- Dette var første gang to programmer kunne ha aksess til de samme dataene samtidig (kvasiparallell aksess)
- IDS ble i 1966 solgt og fra da markedsført som IDMS (Integrated Data Management System)

Maskinvare i 1964

- En stor maskin (1 mill. 1964-$) hadde
  - 512 Kbyte RAM
  - 50 Mbyte disk
  - Annet I/O-utstyr (magnet- og papirbånd, hullkortleser og linjeskriver)
- En slik maskin var vannkjølt, krevde stor plass (2–300 m^2), brukte mye strøm (ca 50 kW), og en stab på 10-12 personer for å holde den i gang
- Drøyt 40 år senere har vi iPod ….

Nettverksdatabaser

- IDS var en *nettverksdatabase*, dvs at skjemaet besto av to typer datastrukturer:
  - Posttyper (COBOL record type)
  - Sett-typer (1:n relasjon mellom to posttyper kalt hhv eier- og medlemstype)
- Hver enkelt post kunne delta i vilkårlig mange sett, som eier i noen og medlem i andre, men bare en gang i hver sett-type
- Topologisk er et nettverksdatabaseskjema en rettet graf med posttypene som noder

Hierarkiske databaser

- Et hierarkisk databaseskjema har to datastrukturer:
  - Posttyper
  - 1:n relasjoner, kalt foreldre–barn-relasjoner, mellom to posttyper
- Foreldre–barn-relasjonene danner hierarkier (trær)
- I realiteten finnes bare ett kommersielt hierarkisk DBMS, IMS, som ble utviklet av IBM og Rockwell International (North American Aviation) og lansert av IBM i 1968
- 95% av *Fortune-1000*-firmaene bruker fortsatt IMS

Relasjonsdatabaser

- Relasjonsdatabaser har én type datastruktur, relasjoner (tabeller), som tilsvarer posttyper
- Kolonnene har navn og kalles attributter
- Linjene er navnløse og kalles tupler (forekomster)
- Tuplenes attributtverdier er atomiske (COBOLs repeterende grupper er ***ikke*** tillatt)
- Alle logiske sammenhenger mellom relasjoner er basert på verdilikhet (fremmednøkler, join)
- SQL er ISO-standard for definisjon og bruk av relasjonsdatabaser

Objektdatabaser

- Dataelementene er objekter
- Objektene kan stå i relasjon (relationship) til hverandre; slike relasjoner er alltid toveis
- Gjeldende standard er ODMG 3.0, men ingen har til nå laget en full implementasjon av denne
- Implementasjonsmessig er objektdatabaser en generalisering av nettverksdatabaser, mens spørre-språket OQL er en beregningskomplett SQL-etterligning (OQL returnerer et objekt; SQL returnerer en relasjon)

2-skjemaarkitektur

- IDS var en 2-skjemaarkitektur database
  - Ett *skjema* ga en fullstendig beskrivelse av databasen
  - Ett *subskjema* tilpasset applikasjonen fungerte som applikasjonens vindu mot databasen
- Hver database hadde bare ett skjema, men kunne ha vilkårlig mange subskjemaer

Datauavhengighet

- 2-skjemaarkitekturen tilbød *datauavhengighet*:
  - Skjemaet definerte både struktur og lagringsformat for dataene
  - Subskjemaet definerte navn og format på de dataene applikasjonen brukte
  - Det ble dermed mulig å forandre lagrings-formatet uten å forandre programmene

3-skjemaarkitektur

- I 1971 foreslo CODASYL DBTG (Committee on Data Systems Languages, Data Base Task Group) å splitte det sentrale skjema i to:
  - Et logisk (begrepsmessig) skjema
  - Et fysisk (internt) skjema
- ANSI/SPARC (ANSI Standards Planning And Requirement Committee) foreslo dette som standard i 1978
- Det begrepsmessige skjema ble ISO-standard i 1982 (sub-skjemaene ble her kalt eksterne skjemaer)
- Vi fikk dermed to former for datauavhengighet:
  - Fysisk mellom internt og begrepsmesig skjema
  - Logisk mellom begrepsmessig og eksternt skjema

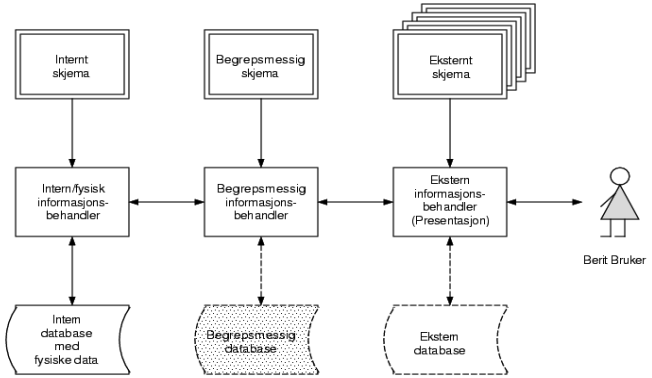
3-skjemaarkitekturen

(Kjernepensum)

3-skjemaarkitekturen for databaser

- **Presentasjonslaget**
  beskrives med eksterne skjemaer (”**views**”) – hvordan informasjon skal presenteres for ulike brukere
- **Det konseptuelle (eller logiske) laget**
  beskrives i det begrepsmessige skjemaet – hva som kan lagres, og hva som er lovlige forandringer
- **Det fysiske laget**
  beskrives i det interne skjemaet – hvordan informasjon lagres, forandres og behandles

3-skjema arkitektur



Fysisk og logisk datauavhengighet

- **Fysisk datauavhengighet**
  betyr at vi kan forandre det interne skjemaet så lenge det ikke strider mot det begrepsmessige skjemaet
  Vi kan f.eks. forandre lagringsformatet av tall fra binært til BCD (Binary Coded Decimal) eller tekst uten å forandre det begrepsmessige skjemaet
- **Logisk datauavhengighet**
  betyr at vi kan beholde eksterne skjemaer selv om vi forandrer det begrepsmessige skjemaet
  Dermed slipper vi å forandre gamle programmer

3-lagsarkitektur i web-applikasjoner

- Benytter ideene fra 3-skjemaarkitekturen i design av distribuerte systemer
  - Presentasjonslag: Webserver
  - Forretningslogikk: Applikasjonsserver
  - Datalag: Databaser, legacysystemer, …