UNIVERSITETET I OSLO

# INF1300
# Introduksjon til databaser

**Forelesere:** Ellen Munthe-Kaas
Ragnar Normann

**Mål:** Kurset skal lære studentene:

- Litt generelt om databaser og databasesystemer med hovedvekt på relasjonsdatabaser
- En metode for design av en relasjonsdatabase for et gitt formål
- Alt om hvordan man lager spørringer mot en relasjonsdatabase



# Kursinnhold

Kurset er delt i to deler:

- Del 1: Databasedesign
- Del 2: Databaseprogrammering

Det blir en midttermineksamen etter del 1
og en avsluttende eksamen etter del 2



# Lærebøker

**Gerhard Skagestein:**

Systemutvikling (2. utgave)
Høyskoleforlaget 2005, ISBN 82-7634-671-5

**Neil Matthew, Richard Stones:**

Beginning databases with PostgreSQL (Second edition)
Apress 2005, ISBN 1-59059-478-9



# Obligatoriske oppgaver

- Det blir fire obligatoriske oppgaver

Innleveringsfrister:

- Oblig1: Onsdag 5. september kl. 12:00
- Oblig2: Onsdag 19. september kl. 12:00
- Oblig3: Onsdag 3. oktober kl. 10:15
  Da blir oppgaven gjennomgått i Store aud.
- Oblig4: Onsdag 14. november kl. 12:00

**Fristene er absolutte, og det blir ikke gitt utsettelse**

# Eksamen

- NB! Ingen læremidler er tillatt brukt på eksamen!
- **Midttermineksamen:**
  Fredag 12. oktober, kl. 9:00–12:00
  Hovedtema: Design av relasjonsdatabaser
- **Avsluttende eksamen**:
  Onsdag 12. desember, kl. 14:30–17.30
  Hovedtema: Bruk av relasjonsdatabaser



# Her begynner vi på pensum



# Filer og databaser



# Transiente og persistente data

- Når vi programmerer, legger vi dataene våre i programvariable (eller bare «variable»)
- Når programmet avsluttes, slettes dataene
- Slike data kalles **transiente**
- Data vi vil ta vare på, er dere vant til å skrive til en fil, det vil vanligvis si til en magnetdisk
- Data som overlever mellom to programkjøringer, kalles **persistente**

# Problemer med filer

- Filer er i allminnelighet tilpasset ett bestemt program
- Uansett kan en fil bare brukes av ett program om gangen (programmet må «åpne» filen for å bruke den, og da er filen sperret for alle andre)
- Store programmer trenger ofte mange typer data, noe som gjerne resulterer i mange filer, noe som igjen gjør programmet enda mer komplisert



# Databaser og DBMS

- En **database** er en samling persistente data som fysisk bare kan aksesseres fra ett program, kalt et **DBMS** (Data Base Management System)
- Vilkårlig mange programmer og brukere kan samtidig være koblet opp mot DBMSet og via det få lest, skrevet og endret data i databasen
- Et slikt program kalles ofte en klient
- Logisk sett kan følgelig vilkårlig mange klienter aksessere dataene i databasen samtidig



# Transaksjoner

- En henvendelse fra en klient til DBMS kalles en **transaksjon**
- En transaksjon som bare leser data, kalles en lesetransaksjon eller en **spørring** (**query**)
- En transaksjon som legger til nye data eller forandrer eksisterende data, kalles en skrivetransaksjon



# Litt historie

- Det første DBMS ble laget og presentert 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 som er en forenkling av nettverksdatabaser
- IMS ble nær enerådende for administrativ data-behandling, og fortsatt er det svært mange store firmaer som har legacy-systemer 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 DBMS for
- Først i 1977 klarte Oracle å lage et DBMS som fortjener betegnelsen relasjonell
- Relasjonsdatabaser er nå svært mye brukt, og de er hovedtemaet for dette kurset



# Informasjon og data



# 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



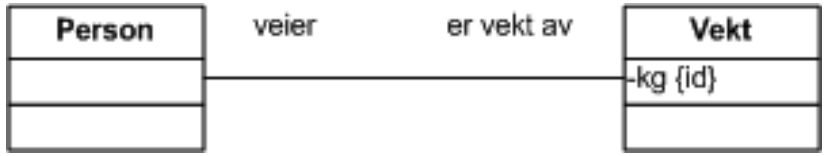
# Informasjon

- **Informasjon** består både av data og 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?

# Vekteksemplet i UML

- I et (nokså ufullstendig) UML klassediagram ser eksemplet ut som nedenfor

  (Vi har ikke med noen attributter for Person, og ikke multiplisiteter, men slike ting skal vi komme tilbake til)



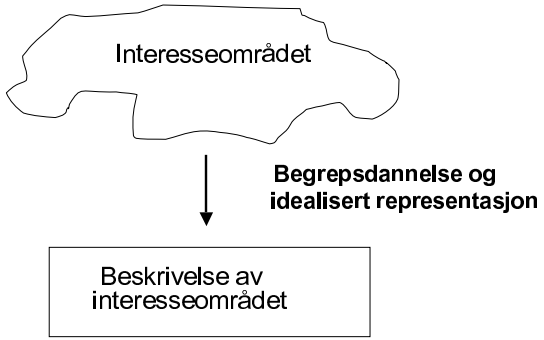



# Interesseområdet (UoD = Universe of Discourse)


- 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 og 100%-prinsippet

- En fullstendig beskrivelse av interesseområdet kalles en **informasjonsmodell**
- *100%-prinsippet* sier at det er mulig å lage en (endelig) informasjonsmodell på norsk
- Informasjonsmodellen uttrykkes gjerne i et *modellspråk*
- Noen aktuelle modellspråk er
  - UML (Unified Modelling Languages)
  - ORM (Object-Role Modelling)
  - ER (Entity Relationship)

# Skranker

- Beskrivelsen av forretningsreglene kalles **skranker**
- *Statiske skranker* beskriver begrensninger på mulige tilstander i interesseområdet
- *Dynamiske skranker* beskriver begrensninger på mulige forandringer i interesseområdet
- I dette kurset skal vi konsentrere oss om statiske skranker



# Det begrepsmessige skjema

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

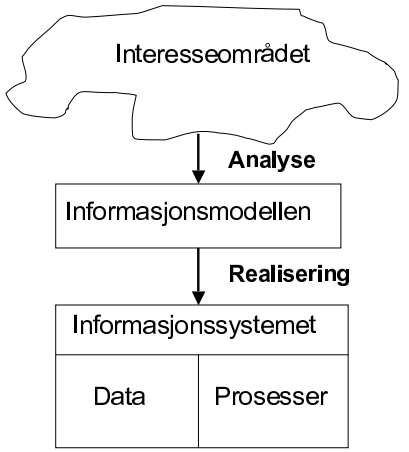


# Integritetsregler

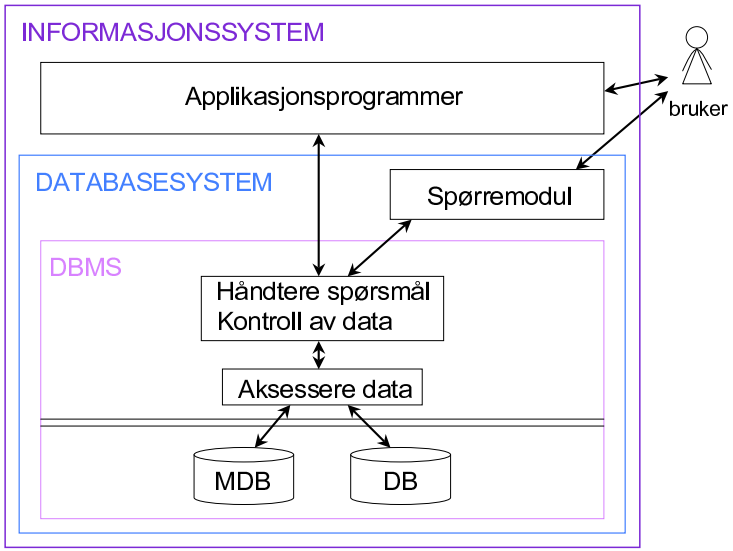
- I det begrepsmessige skjemaet kaller vi skrankene for **integritetsregler**
- Integritetsreglene bestemmer
  - hva som er lovlig å *lagre* i informasjons-systemet (lovlige tilstander i databasen)
  - hva som er lovlige *forandringer* (lovlige transisjoner/transaksjoner)
- Det er umulig for en bruker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



# Informasjonssystemer



Informasjonssystemer vs. databaser






# DBMS – Database Management System

- Spesialisert programvare som understøtter:
  - Persistens
  - Transaksjonshåndtering
    La *t* være en transaksjon. Da gjelder:
    - **A** tomicity (alt eller intet av *t* blir gjort)
    - **C** onsistency (*t* kan ikke gjøre DBen inkonsistent)
    - **I** solation (*t* merker ikke noe til andre transaksjoner)
    - **D** urability (endringer gjort av *t*, er varige)
  - Programmeringsgrensesnitt

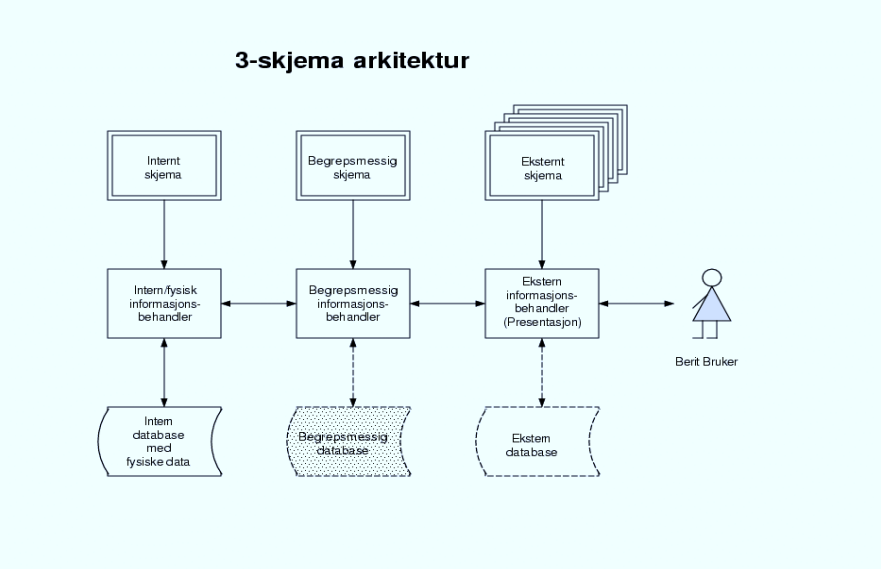


# 3-skjemaarkitekturen



# 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-lagsarkitektur

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



# Begreper og representasjon



# Begreper

- Helt fra menneskene fikk språket, har vi satt navn på grupper av tilsvarende ting
- Platon, i sin idé-lære, var den første som beskrev dette systematisk (vitenskapelig)
- Selv om det ikke er noen kyr her inne, vet dere alle hva jeg mener med ordet «ku»
- Og selv de som aldri har sett (et bilde av) en klapperslange, forstår ordet «klapperslange»
- *Ku* og *klapperslange* er to eksempler på begreper
- To mer abstrakte eksempler er *lån* og *flyavgang*

# Representasjon

- For å lage et begrepsmessig skjema for UoD må vi velge hvilke begreper skjemaet skal inneholde
- I tillegg må vi for hvert begrep bestemme oss for hvordan vi skal lagre informasjon om forekomster av dette begrepet
- Vi kan ikke lagre kyr i databasen, så vi (eller bonden som skal registrere melkeproduksjon) må finne en måte å lagre informasjon som identifiserer hver enkelt ku
- En slik identifikasjonsmåte kalles en **representasjon** eller **identifikator** for begrepet

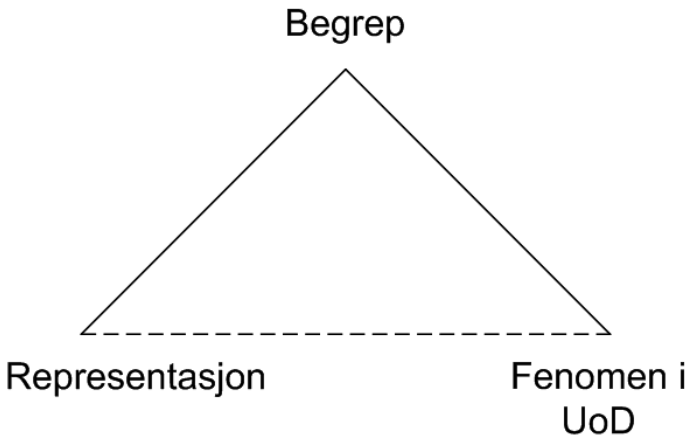


# Ontologi

- 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





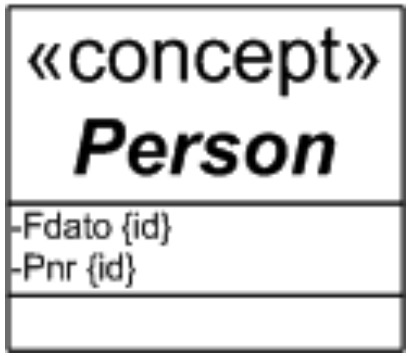

# Begreper i UML

- Standard UML har ikke noe symbol for begreper
- Men UML tillater spesialisering av sine symboler
- En slik spesialisering kalles en **stereotypi** (Engelsk: stereotype) som betegnes med et ord i guillemets («»-parenteser)
- Vi definerer stereotypien **«concept»** som en klasse som ikke får lov til å ha andre attributter enn sin representasjon (identifikator)
- På norsk kaller vi en «concept»-klasse et **begrep**

# Eksempel på et UML-begrep

- Til høyre står UML-symbolet for begrepet Person
- Representasjonen er sammensatt av
  - Fdato (fødselsdato)
  - Pnr (personnummer)

  på henholdsvis 6 og 5 sifre (det er det vi kaller Fnr (fødselsnummer))






# ORM-UML

- Et UML klassediagram hvor alle klassene er begreper, kalles et ORM-UML-diagram
- ORM står for Object-Role-Modeling, og er den anbefalte modelleringsmetoden i Microsofts .Net
- Selv om de grafiske symbolene er forskjellige, er ORM og ORM-UML nær identiske modellerings-metoder



# Neste uke ....

- .... ser vi på sammenhengen mellom ORM-UML og vanlig språk
- Vi begynner å lage enkle modeller og ser på
  - Assosiasjoner mellom begreper
  - Entydighet og multiplisitet
  - Funksjonelle avhengigheter
  - Overgang fra ORM-UML til vanlig UML og til relasjonsdatabaseskjemaer