IN2090 – Databaser og datamodellering

01 – Introduksjon og motivasjon: Databaser

Leif Harald Karlsen
leifhka@ifi.uio.no

Universitetet i Oslo

Hvorfor bruke databaser?

Motivasjon

Hvorfor bruke databaser?

Hvorfor ikke bare bruke f.eks. Python-lister eller Java Collections?

```
handlekurv = ["gulrot", "ost", "juice", "melk"]
```

1. Dataenes levetid
   - Variabler blir lagret i RAM (Random Access Memory)
   - Dette minnet blir slettet hver gang maskinen slås av
   - Vil ofte bevare data igjennom omstart
2. Skalerbar lagringsplass
   - 1 GB harddisk-minne mye billigere enn 1 GB med RAM
3. Separere data fra kode
   - Pythons data er kun tilgjengelig fra Pythons kjøretid
   - Vil ofte la dataene være tilgjengelig for flere programmer

Alle disse problemene er løst av filsystemet!

Motivasjon

Hvorfor bruke databaser?

Så hvorfor ikke bare bruke filer?

Python + Filer

```
import csv
import os

filea = "a.csv"
fileb = "b.csv"
temp = "temp.csv"
source1 = csv.reader(open(filea,"r"),delimiter=",")
source2 = csv.reader(open(fileb,"r"),delimiter=",")

source2_dict = {}
for row in source2:
    source2_dict[row[0]] = row[1]

with open(temp, "w") as fout:
    csvwriter = csv.writer(fout, delimiter=delim)
    for row in source1:
        if row[1] in source2_dict:
            row[3] = source2_dict[row[1]]
        csvwriter.writerow(row)
os.rename(temp, filea)
```

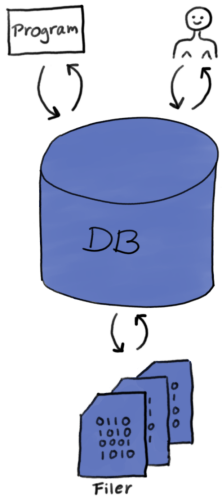
SQL + Database

```
UPDATE a
   SET c4=b.c2
  FROM b
 WHERE a.c2 = b.c1;
```

Motivasjon

Hvorfor bruke databaser?

Så hvorfor ikke bare bruke filer?

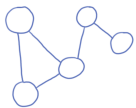


Databasen fungerer som et abstraksjonslag over filsystemet

- Enklere å søke i og manipulere data
- Enklere å spesifisere strukturen på dataene
- Effektivitet og skalerbarhet

Databaser

- En database er en samling data på et bestemt format
- Ulike typer databaser som fokuserer på lagring og håndtering av ulike typer data
- **Dokument-databaser:** Fokuserer på dokumenter og effektive søk i store mengder tekst
- **Key-value stores:** Fokuserer på nøkkel-verdi-par (JSON, Dictionaries, HashMaps, osv.)
- **Grafdatabaser:** Fokuserer på grafer og effektivt finne stier mellom noder i en graf
- **Relasjonelle databaser:** Fokuserer på data i tabell-form (mengder av tupler)
- I dette kurset skal vi kun jobbe med relasjonelle databaser



Relasjonelle databaser: Forenklet beskrivelse

- En relasjonell database er en samling tabeller
- En tabell er også kalt en relasjon
- En tabell har
  - et navn,
  - en samling kolonner
  - og en samling rader (som er dataene)
- En kolonne har
  - et navn,
  - og en type

Customers

CustomerID (int)	Name (text)	Birthdate (date)	NrProducts (int)
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

Relasjonelle databaser = Regneark?

Så, relasjonelle databaser er egentlig bare regneark?

Nei, i motsetning til regneark har relasjonelle databaser:

- en rigid struktur
- spørrespråk for uthenting og manipulering av data
- programmatiske grensesnitt for interaskjon med databasen (fra f.eks. Python eller Java)
- systemer for sikkerhet og kontroll av hvem som har tilgang til dataene
- systemer som sikrer integritet av dataene
- støtte for langt større og mer kompliserte datamengder

Databasesystemer

- Egentlig er en database bare en mengde data (ikke et system/program)
- Et databasesystem (DBMS) er
  et system som lar brukere definere, lage, vedlikeholde og kontrollere tillgangen til databasen.
- Et relasjonelt databasesystem (RDBMS) er
  et databasesystem over relasjonelle databaser.
- Bruker ofte ordet “database” for både dataene, programmet, og kombinasjonen

Hvorfor Relasjonelle Databaser?

- Relasjonelle databaser er den desidert mest brukte typen database
- Nesten all data kan (naturlig) representeres som tabeller
- Naturlig format å jobbe med
- Enkelt å presist spesifisere strukturen til dataene (metadata)
- Denne rigide strukturen tillater svært effektiv uthenting, manipulering og oppdatering av data
- Gir også mange ulike typer sikkerhet

Når bør man ikke bruke relasjonelle databaser?

Finnes også tilfeller når man ikke bør bruke relasjonelle databaser, f.eks. når:

- dataene ikke har noen klar struktur (f.eks. ren tekst, video, lyd)
  - Bruk dokument-databaser
- man er interessert i hvilke relasjoner gitte individer har (*"hva er relasjonen mellom Ola og Kari?"*) fremfor hvilke individer som har hvilke gitte relasjoner (*"hvem er mor til Ola?"*)
  - Bruk grafdatabaser
- man alltid må lese store, men kjente mengder data inn i minne ved hver bruk (f.eks. konfigurasjonsfiler, grafikk)
  - Bruk vanlige filer

Spørrespråk: SQL

- Et spørrespråk et et språk for å formulere spørringer (spørsmål) til en database
- De fleste spørrespråk er *deklarative*, man uttrykker *hva* man vil finne fremfor *hvordan* det skal finnes
- SQL er det mest brukte spørrespråket og det vi skal lære i dette kurset
- SQL har også funksjonalitet for manipulering (oppdatere, sette inn, slette, osv.) av data og metadata
- SQL kan enten skrives og kjøres direkte av mennesker, eller bli generert og kjørt av programmer (f.eks. Java eller Python)

```
SELECT age
  FROM person
 WHERE name = 'Kari';
```

SQL-spørring som finner alderen til personen med navn Kari.

Takk for nå!

Neste video handler om datamodellering.