

IN2090 – Databaser og datamodellering

01 – Introduksjon og motivasjon

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

- ◆ Pythons data 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?

- ◆ Enkelere å manipulere data
 - ◆ Enklere å finne, sette inn, slette og oppdatere data
 - ◆ Håndterer lesing og skrivning fra flere programmer samtidig
- ◆ Spørrespråk
 - ◆ Enklere å fortelle maskinen *hva* du ønsker å beregne (hva skal oppdateres, hva man ønsker å vite, osv.), fremfor *hvordan* man skal beregne det
- ◆ Effektivitet
 - ◆ Databaser bruker avanserte teknikker for å finne den mest effektive måten å finne svaret på spørringer
 - ◆ Bruker også avanserte datastrukturer for å lagre dataene på en måte som gir effektiv uthenting

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.)
- ◆ Graf-databaser: 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

En (forenklet) beskrivelse av relasjonelle databaser:

- ◆ 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

Tabeller/Relasjoner

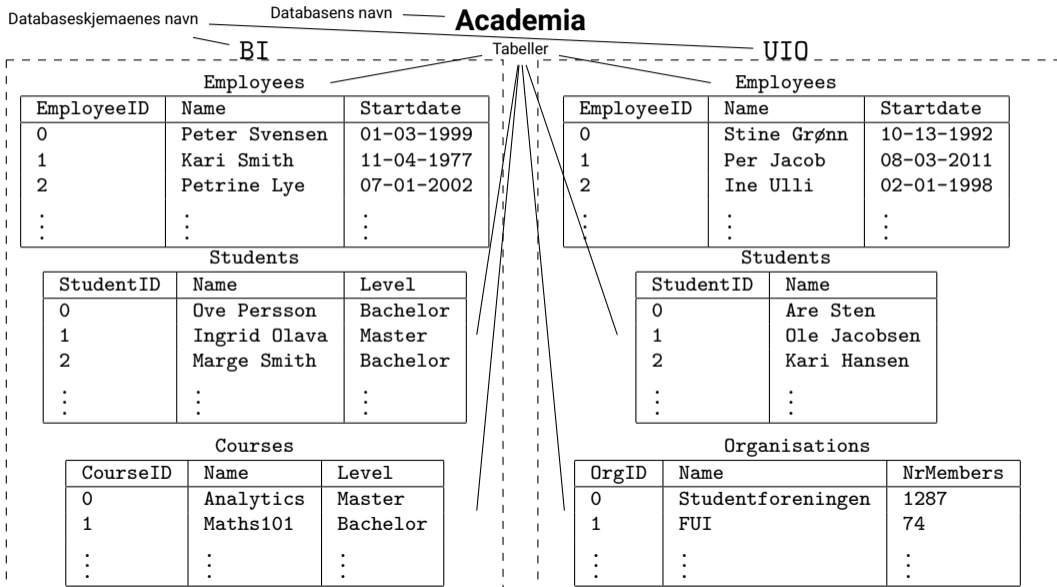
Eksempel tabell:

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

Vi dropper ofte å skrive typene i kolonnene.

Eksempel Database



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
- ◆ programmatisk grensesnitt for interaksjon 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 tilgangen til databasen.
- ◆ Databasesystemet er altså *programmet* som håndterer *dataene/databasen*
- ◆ Et relasjonelt databasesystem (RDBMS) er
et databasesystem over relasjonelle databaser.
- ◆ Men, bruker ofte ordet “database” for både dataene, programmet, og kombinasjonen av de to

Databasesystemet sørger for at

- ◆ dataene til enhver tid er i henhold til metadataene (skjemaet)
- ◆ spørringer blir besvart på den mest effektive måten
- ◆ sikkerheten og konfidensialiteten til dataene ivaretaes
- ◆ integriteten til dataene ivaretaes

Hvorfor Relasjonelle Databaser

- ◆ Relasjonelle databaser er den desisert 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:

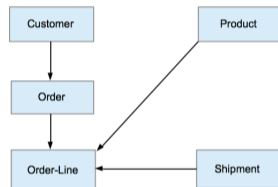
- ◆ dataene ikke har noen klar struktur (f.eks. ren tekst, video, lyd)
 - ◆ Bruk dokument-databaser
- ◆ man er interessert i hvilke relasjoner gitte individer har (*“hva er relasjonen mellom Ola og Kari?”*) i motsetning til hvilke individer som er i hvilke gitte relasjoner (*“hvem er mor til Ola?”*)
 - ◆ 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

- ◆ Et spørrespråk er et språk for å formulere spørringer (spørsmål) til en database
- ◆ De fleste spørrespråk er *deklarative*
 - ◆ man uttrykker *hva* man ønsker å vite fremfor *hvordan* man skal finne svaret
 - ◆ I Python og Java forteller man datamaskinen *hvordan* man skal beregne svaret
 - ◆ I deklaratve språk sier man altså ikke hvordan, bare *hva* man ønsker svar på
 - ◆ Det er så opp til databasesystemet å finne ut *hvordan* spørringen skal besvares
- ◆ 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-spørringer kan enten skrives og kjøres direkte av mennesker, eller bli generert og kjørt av programmer (f.eks. Java eller Python)

Litt historie

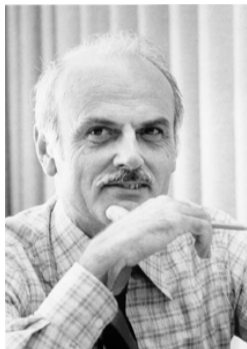
- ◆ Det første DBMSet ble utformet av Bachman og Williams i 1964
- ◆ Ble solgt under navnet IDMS (*Integrated Database Management System*)
- ◆ IDMS var en *nettverksdatabase* og var designet for bruk fra et programmeringsspråk
- ◆ I 1968 kom IBM med IMS, en *hierakisk database* som forenkling av nettverksdatabaser
- ◆ IMS var nær enerådende for administrativ databehandling
- ◆ Fortsatt svært mange store firmaer som har legacy-systemer som bruker IMS



Eksempel
nettverksdatabaseskjema

Litt mer historie

- ◆ I 1970 presenterte E. F. Codd sin *relasjonsmodell*
- ◆ Dette var den teoretiske beskrivelsen av den da nye typen databaser kalt *relasjonsdatabaser*
- ◆ Relasjonsdatabaser er enkle å beskrive og bruke, men vanskelig å lage DBMS for
- ◆ Først i 1977 klarte Oracle å lage et DMBS som fortjener betegnelsen *relasjonell*
- ◆ Relasjonsdatabaser har vært svært mye brukt i mange tiår nå



Edgar Frank "Ted" Codd

Hvorfor datamodellering?

Data vs. informasjon

- ◆ Innholdet i en database er *data*
- ◆ F.eks. i en kolonne som heter *Vekt* kan man ha verdien 142.3
- ◆ Dette er en bit data, og sier oss egentlig ingenting!
- ◆ Må vite hvordan verdien skal tolkes for at det skal gi oss noe *informasjon*
- ◆ F.eks. hvilken måleenhet er brukt, hva er det vekten på, betegner det faktisk vekt eller maks vekt, osv.
- ◆ Informasjon er data pluss regler for hvordan data skal tolkes

Heis	Vekt	...
⋮	⋮	⋮
3	142.3	...
⋮	⋮	⋮

Heis nr. 3 veier 142.3kg?

3 heiser har maks kapasitet 142.3 kg?

Heis nr. 3 har maks kapasitet 142.3 tonn?

Heiser kan ta 3 mennesker som hver veier 142.3 kg?

Fra domene til data



- ◆ En database beskriver et domene
- ◆ Så vi må oversette informasjon fra domene til data
- ◆ Modellering hjelper med denne prosessen

Hvorfor datamodellering?

Kompliserte domener: Entiteter

De fleste domener er veldig komplekse!

Universitet:

studenter
ansatte
kurs
foreninger
programmer
bygninger
rom
aktiviteter
forskningsgrupper
karakterer
prosjekter
⋮

Nettbutikk:

produkter
kunder
bestillinger
leverandører
lagere
kategorier
kampanjer
regioner
relatert
kvitteringer
dokumenter
⋮

Sykehus:

pasienter
ansatte
medisiner
sykdommer
behandlinger
rutiner
krav
rom
operasjoner
inventar
vakt
⋮

Skatt:

personer
lønnstrinn
relasjoner
frynsegoder
bedrifter
organisasjoner
sykemeldinger
familiære forhold
formue
arv
fagforeninger
⋮

Hvorfor datamodellering?

Kompliserte domener: Relasjoner

Disse *entitenene* har mange relasjoner mellom seg

Universitet:

ansatt ved
rom bestilt av
veileder
har karakter i
ledes av

⋮

Nettbutikk:

bestilt av
har kategori
koster
er med i kampanje
antall på lager

⋮

Sykehus:

får behandling
har kompetanse til
har vakt
har utstyr
trinn i rutine

⋮

Skatt:

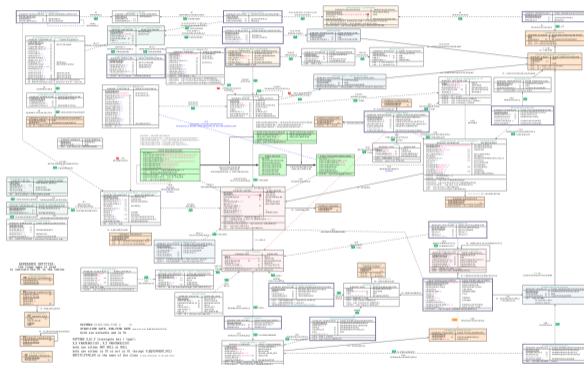
er datter av
tjener
medlem av
ansatt i
har lønnstrinn

⋮

Hvorfor datamodellering?

Kompliserte databaser

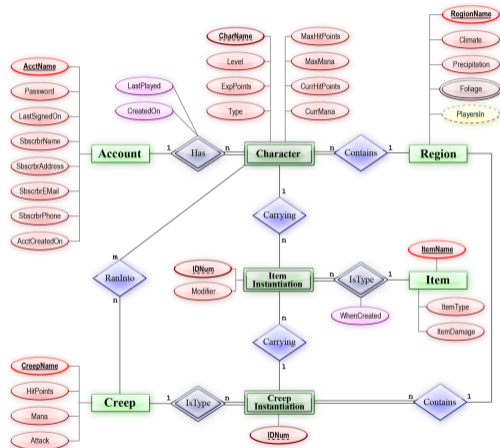
- ◆ Så databasen blir da veldig kompleks
 - ◆ Mange tabeller (f.eks. > 100)
 - ◆ Mange kolonner (f.eks. > 20)
 - ◆ Mange rader (millioner)
- ◆ Må da ha en god metode for å designe skjemaet til databasen
- ◆ Altså, hvilke tabeller og kolonner man skal ha, og hvordan tabellene er relatert



Komplisert databaseskjema

Datamodellering

- ◆ Et modelleringsspråk brukes for å lage en modell av et domene
- ◆ En modell er en forenkling av virkeligheten som beskriver kun de tingene vi er interessert i
- ◆ Denne modellen blir så oversatt til et databaseskjema
- ◆ Finnes mange modelleringsspråk:
 - ◆ UML
 - ◆ ORM
 - ◆ ER
 - ◆ OWL
- ◆ Vi skal bruke ER



ER-diagram av karakterer for et spill¹

¹https://commons.wikimedia.org/wiki/File:ER_Diagram_MMORPG.png

Eksempelanvendelse

Eksempelanvendelse

Beskrivelse

- ◆ La oss si at vi skal lage et kinobillett-system
- ◆ Programmet skal kunne:
 - ◆ La ansatte legge inn info om saler, filmer, visninger, priser, osv.
 - ◆ La kunder bestille billetter, se program, osv.
 - ◆ Lage rapporter over billettsalg ol. for kinoledelsen
 - ◆ Tillate analyse over dataene



Interessedomene

Eksempelanvendelse

Fra domene til modell (1)

1. Bestemme hvilke ting fra domene vi er interessert i:

- ◆ Kino
- ◆ Filmer
- ◆ Kinosaler
- ◆ Billetter
- ◆ Visninger
- ◆ Priser
- ◆ Program
- ◆ ...



2. Bestemme hva som er sant i vårt tilfelle

- ◆ Kan det være mer enn én sal per kino?
- ◆ Kan én billett brukes til å se mer enn én visning?
- ◆ Er det alltid én billett per person, eller finnes det gruppebilletter?
- ◆ Er det faste plasser på billetten?
- ◆ ...

Eksempelanvendelse

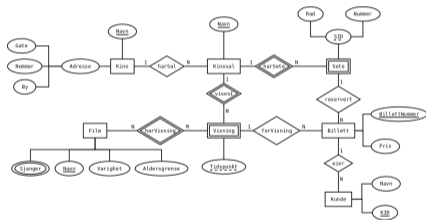
Fra domene til modell (2)

2. Bestemme hva som er sant i vårt tilfelle

- ◆ Kan det være mer enn én sal per kino?
- ◆ Kan én billett bruker til å se mer enn én visning?
- ◆ Er det alltid én billett per person, eller finnes det gruppebilletter?
- ◆ Er det faste plasser på billetten?
- ◆ ...



3. Lage modell i henhold til disse faktaene

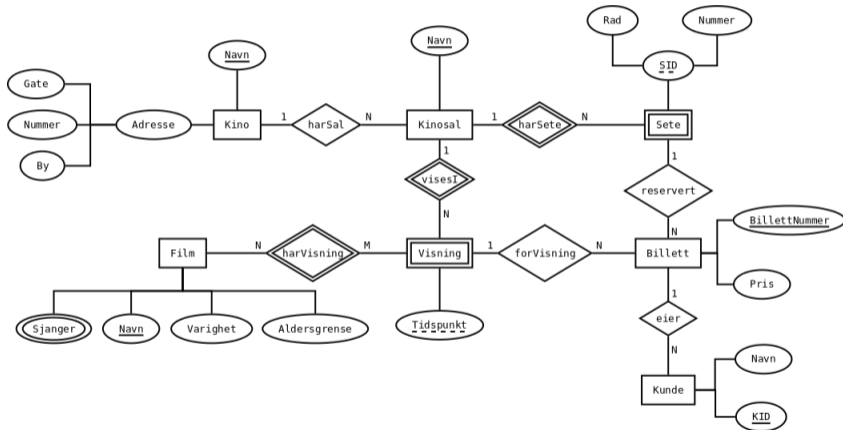


ER-diagram som modellerer domene vårt

Eksempelanvendelse

Fra domene til modell (3)

3. Lage modell i henhold til disse faktaene

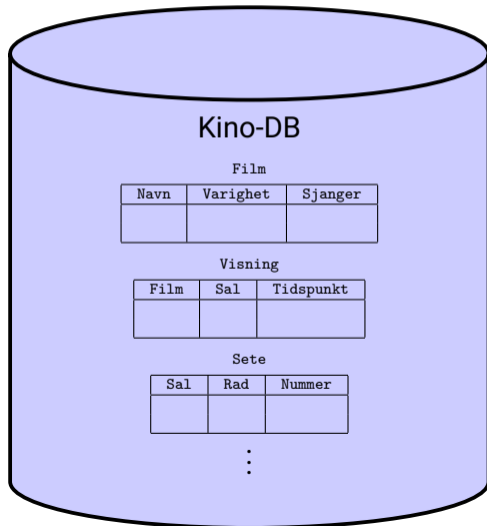
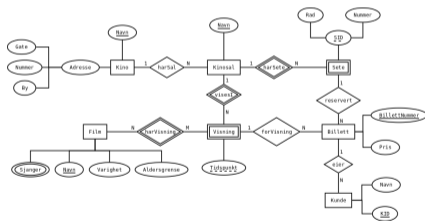


ER-diagram som modellerer domene vårt

Eksempelanvendelse

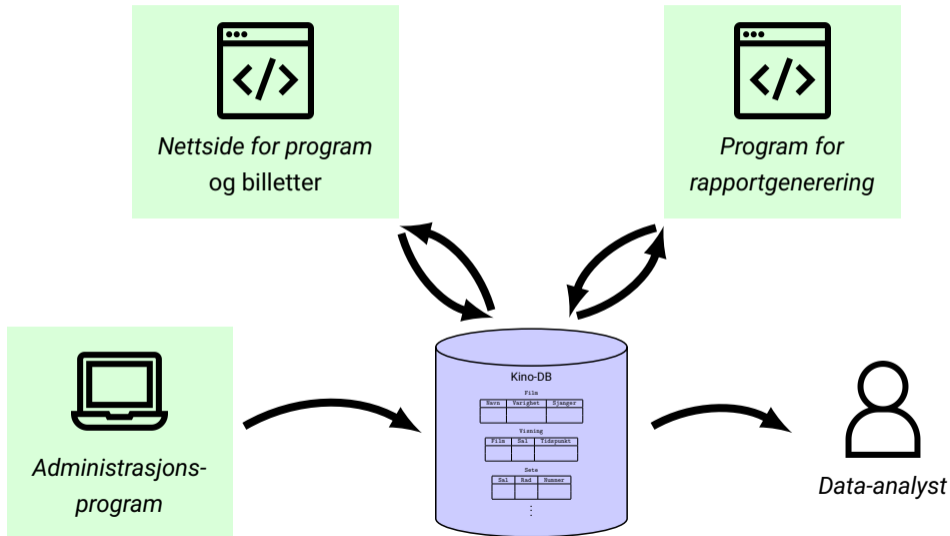
Fra modell til databaseskjema

4. Lage databaseskjema i henhold til modell



Eksempelanvendelse

Fra databaseskjema til bruk



Praktisk informasjon

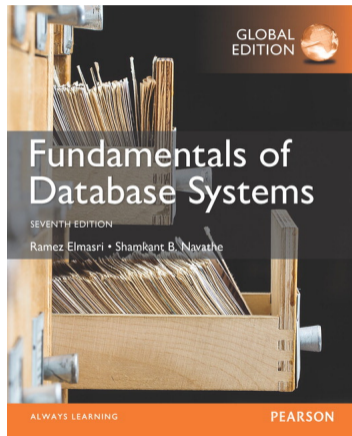
Oversikt over kurset

No.	Tema	Dato	Obliger
1	Introduksjon og motivasjon	Man 19.08	
2	<i>Modellering</i> : Relasjonelle modellen	Man 26.08	
3	<i>Modellering</i> : ER-modellering 1	Man 02.09	
4	<i>Modellering</i> : ER-modellering 2	Man 09.09	1. Modellering
5	SQL: Grunnleggende SQL	Man 16.09	
6	SQL: Enkle joins og nestede SELECT	Ons 18.09	2. Enkel SQL
7	SQL: Datamanipulering og views	Man 23.09	
8	SQL/Modellering: Typer og skranker	Ons 25.09	
9	<i>Modellering</i> : Normalformer 1	Man 30.09	
10	<i>Modellering</i> : Normalformer 2	Ons 02.10	3. Normalformer
11	SQL: Aggregering og sortering	Man 07.10	
12	SQL: Ytre joins og mengde operatorer	Man 14.10	4. Kompleks SQL
13	SQL: Programmering med databaser	Man 04.11	5. Programmering med SQL
14	SQL: Sikkerhet	Ons 06.11	
15	SQL: Indekser og spørreprosessering	Man 11.11	
16	<i>Modellering</i> : Repetisjon	Man 18.11	
17	SQL: Repetisjon	Ons 20.11	

Oversikt over kurset

- ◆ To hovedkategorier: Modellering og SQL
- ◆ Kan være noen av de ubrukte onsdagene blir brukt til ekstra repetisjon
- ◆ Blir et opphold på to uker i oktober grunnet reise, bruker da gruppetimer til repetisjon av vanskelige tema

- ◆ Foilene til kurset
 - ◆ Publisert på timeplanen på semestersiden
 - ◆ Screencast med lyd blir også publisert
- ◆ Ukesoppgavene og obligene
 - ◆ Gjennomgås på gruppetimene
- ◆ Boken *Fundamentals of Database Systems*
 - ◆ Tykk bok, men skal bare bruke deler av den
 - ◆ Se timeplanen for hva som er pensum
 - ◆ Gjenbrukes i andre kurs (IN3020)



- ◆ Fem obliger
- ◆ Leveres via Devilry
- ◆ Dersom du blir syk el., kan man be om utsettelse
 - ◆ Vi (forelesere og gruppelærere) kan gi intill 3 dager utsettelse
 - ◆ Administrasjonen håndterer lengre utsettelse (krever legeattest)

- ◆ RDBMS: PostgreSQL 11²
 - ◆ Avansert relasjonell databasesystem
 - ◆ Åpen kildekode og støtter de fleste plattformer
- ◆ Modelleringsverktøy: Dia³ + Erdtosql⁴
 - ◆ Dia er et generelt diagram-verktøy
 - ◆ Vi skal bruke det med *ER Sheet* (mer om dette siden)
 - ◆ Åpen kildekode og støtter de fleste plattformer
 - ◆ Erdtosql har jeg skrevet, verifiserer og oversetter modeller til databaseskjema
- ◆ Databaser:
 - ◆ IMDB: Filmdatabase
 - ◆ Northwind: Database over produkter, bestillinger, osv.

²<https://www.postgresql.org/download/>

³<http://dia-installer.de/download/index.html>

⁴<https://github.com/uio-no/IN2090/erdtosql>

PostgreSQL





Leif Harald Karlsen

leifhka@ifi.uio.no



Dumitru Roman

dumitrur@ifi.uio.no



Envgenij Thorstensen

evgenit@ifi.uio.no

Gruppelærere og rettere

Gruppelærere:

- ◆ Ahmed Abbas
- ◆ Henrik Askjer
- ◆ Julia Winsevik Haugen
- ◆ Magne Svendsen
- ◆ Benedikte Solstad
- ◆ Justyna Ozog
- ◆ Gudmundur Jonsson

Rettere:

- ◆ Kaja Ivarsøy
- ◆ Tan Van Ngoc Tran
- ◆ Alexandra Huynh

- ◆ Vi bruker Piazza²
- ◆ Et forum for diskusjon av pensum, oppgaver, osv.
- ◆ Dere skal alle få en mail med info i løpet av denne uken
- ◆ (Hvis ikke du får noen mail, send meg en mail)

²<https://piazza.com/>

Nyttige lenker (Ikke pensum)

- ◆ Mer om databaser:
 - ◆ Engelsk: <https://en.wikipedia.org/wiki/Database>
 - ◆ Norsk: <https://no.wikipedia.org/wiki/Database>
- ◆ Mer om relasjonelle databaser:
 - ◆ Engelsk: https://en.wikipedia.org/wiki/Relational_database
 - ◆ Norsk: <https://no.wikipedia.org/wiki/Relasjonsdatabase>
- ◆ Mer om datamodellering:
 - ◆ Engelsk: https://en.wikipedia.org/wiki/Data_modeling
 - ◆ Engelsk: https://en.wikipedia.org/wiki/Data_model
- ◆ Mer om ulike typer databaser (engelsk):
<https://dzone.com/articles/the-types-of-modern-databases>
- ◆ *What is the benefit of learning SQL?* (engelsk):
<https://www.quora.com/What-is-the-benefit-of-learning-SQL>