INF1300 Introduksjon til databaser: SQL—Structured Query Language

En første introduksjon
Lysark til forelesning onsdag 21. september 2011

UNIVERSITETET I OSLO

Dagens tema

- SQLs definisjonsspråk
- SQLs spørrespråk
  - select-from-where
  - distinct
  - order by
- SQLs manipulasjonsspråk
- Indekser
- PostgreSQL

SQL—Structured Query Language

- **SQL—Structured Query Language**—er et deklarativt språk for spørringer mot relasjonsdatabaser
- SQL inneholder også konstruksjoner for å definere nye relasjonsdatabaser, for å legge inn og endre data i databasen, mm
- En referanse til PostgreSQL er http://www.postgresql.org/docs/8.2/interactive/

Hvordan uttale «SQL»?

- På norsk uttaler vi SQL bokstav for bokstav: «ess-ku-ell»
- På engelsk uttales SQL «'si:kwel»
  Årsaken er historisk:
  - SQL ble utviklet av IBM, og prototypen het SEQUEL—et akronym for «Structured English QUEry Language»
  - Da SEQUEL ble lansert som et produkt i 1976, ble navnet forkortet til SQL, men uttalen ble beholdt og har overlevd til nå

SQL-standarder

- Flere standarder:
  - ANSI SQL (1986)
  - SQL2 (SQL-92)
  - SQL3 (SQL:1999) = SQL2 + objekt-relasjonelle egenskaper mm
- Mange dialekter:
  - Hvert DBMS har sine særegenheter. Sjekk den aktuelle DBMSens dokumentasjon!
  - Alle oppfyller ANSI SQL, stort sett også SQL2
  - Noen deler av SQL er ikke veldefinert (selv ikke i ANSI SQL) og behandles derfor potensielt forskjellig i forskjellige DBMSer

SQLs bestanddeler

SQL består av en samling konstruksjoner som funksjonelt, men ikke syntaktisk, kan deles opp slik:

- SDL: Storage Definition Language
  3-skjema-arkitekturens fysiske lag
- DDL: Data Definition Language
  3-skjemaarkitekturens konseptuelle lag
- VDL: View Definition Language
  3-skjemaarkitekturens presentasjonslag
- DML: Data Manipulation Language
  innlegging, endring og sletting av data
- DQL: Data Query Language—spørrespråk
- DCL: Data Control Language—integritet og sikkerhet

SQLs DDL—Data Definition Language

- **create**: Opprette tabell
- **drop**: Fjerne tabell
- **alter table**: Endre tabell, *herunder:*
  - Legge til eller fjerne kolonner
  - Legge til, fjerne eller endre integritetsregler (constraints)

create

create table R
(A_1 D_1 [S_1],
...
A_n D_n [S_n],
[*liste av skranker*]
);

R er navnet på relasjonen
A_i er et atributt
D_j er et domene
S_k er en skranke

[ ] betyr at dette leddet er en valgfri del av setningen

create—eksempel

Ansatt(AId, Navn, Tittel, Fdato, Pnr, AnsDato)

- Vi ønsker ikke at to ansatte skal kunne ha samme AId
- To personer kan aldri ha samme fødselsnummer = Fdato + Pnr
- Dermed er både AId og (Fdato, Pnr) kandidatnøkler. Vi velger AId som primærnøkkel og markerer (Fdato, Pnr) som kandidatnøkkel.

create—eksempel

Ansatt(AId, Navn, Tittel, Fdato, Pnr, AnsDato)

- Vi ønsker ikke at to ansatte skal kunne ha samme AId
- To personer kan aldri ha samme fødselsnummer = Fdato + Pnr
- Dermed er både AId og (Fdato, Pnr) kandidatnøkler. Vi velger AId som primærnøkkel og markerer (Fdato, Pnr) som kandidatnøkkel.

```
create table Ansatt (
    AId         int primary key,
    Navn        varchar(40) not null,
    Tittel      varchar(15) not null,
    Fdato       char(6) not null,
    Pnr         char(5) not null,
    AnsDato     date,
unique (Fdato, Pnr)
);
```

Datatyper i PostgreSQL

datatype	*forklaring*
`integer`	heltall
`real`	flyttall
`numeric(n,d)`	n desimale sifre, d etter desimal-punktum
`char(n)`	tekst med fast lengde
`varchar(n)`	tekst med variabel lengde
`boolean`	boolsk verdi
`date`	dato
`time`	klokkeslett
`timestamp`	dato og klokkeslett
`bit(n)`	bitstreng med fast lengde
`bit varying(n)`	bitstreng med variabel lengde

Primærnøkler

- Kan deklareres i **create table** sammen med primærnøkkelattributtet (bare hvis attributtet utgjør primærnøkkelen alene)

```
create table Ansatt (
AId             int primary key,
...  );
```

Primærnøkler

- Kan deklareres i **create table** sammen med primærnøkkelattributtet (bare hvis attributtet utgjør primærnøkkelen alene)

```
create table Ansatt (
    AId         int primary key,
... );
```

- Kan deklareres separat i **create table** etter attributtdeklarasjonene

```
create table Ansatt (
    AId         int,
    ...
primary key (AId)
);
```

Primærnøkler, regler

- Maks én primærnøkkeldeklarasjon pr. relasjon
- Konsekvenser av deklarasjonen:
  - To tupler i relasjonen får ikke stemme overens i alle attributtene i primærnøkkelen. Forsøk på brudd ved **insert** eller **update** skal avvises av DBMSet
  - Attributtene i primærnøkkelen får ikke inneholde **null**
- Dette må sjekkes av systemet ved hver **insert** og hver **update**

Kandidatnøkler

- Kan deklareres i **create table** *sammen med nøkkelattributtet* (bare hvis attributtet utgjør kandidatnøkkel alene)

```
create table Ansatt (
    ...
    Fnr     char(11) not null unique,
    ... );
```

Kandidatnøkler

- Kan deklareres i **create table** *sammen med nøkkelattributtet* (bare hvis attributtet utgjør kandidatnøkkel alene)

```
create table Ansatt (
    ...
    Fnr     char(11) not null unique,
    ... );
```

- Kan deklareres separat i **create table** *etter* attributtdeklarasjonene

```
create table Ansatt (
    ...
    Fdato   char(6) not null,
    Pnr     char(5) not null,
    ...
unique (Fdato, Pnr)
);
```

Kandidatnøkler, regler

- Flere kandidatnøkkeldeklarasjoner er tillatt pr. relasjon
- Konsekvenser av deklarasjonen:
  - To tupler i relasjonen får ikke stemme overens i alle attributtene i kandidatnøkkelen
  - Kan brytes hvis ett eller flere av attributtene i kandidatnøkkelen inneholder **null**
- Dette må sjekkes av systemet ved hver **insert** og hver **update**

Skranker

- Skranke på et attributt eller et tuppel i en relasjon:
  create table R (... **check** ...);
  - Sjekkes ved **insert** og **update** av R
- Skranke på tvers av relasjoner:
  create trigger T ...;
  - Triggere «vekkes» når en hendelse (typisk insert, update, delete på en relasjon) skal til å inntreffe
  - Når triggerne «vekkes», eksekveres en tilhørende metode

Skranke på ett attributt

- not null
  - **create table** Ansatt (**...** Fdato int **not null**, **..**);
  - Konsekvenser:
    - Kan ikke sette inn tuppel med verdien null i attributtet
    - Kan ikke endre verdien til null senere
- check
  - ```
    create table Ansatt (
        ...
        Tittel varchar(15)
        check (Tittel='Selger' or Tittel='Direktør' or ...),
        ...);
    ```
  - Angir en betingelse på attributtet. Sjekkes ved hver endring av attributtets verdi
    ```

Fremmednøkler

- Deklarasjon av fremmednøkler:

```
create table Timeliste (
    AId      int references Ansatt (AId),
    ...);
```

- Alternativt:

```
create table Timeliste (
    AId      int,
    ...
    foreign key (AId) references Ansatt (AId)
    ...);
```

Fremmednøkler, regler

- Konsekvenser av deklarasjonen:
  - De refererte attributtene må være deklarert **unique** eller **primary key**
  - Verdier (≠ **null)** som opptrer i fremmednøkkelens refererende attributter *må* opptre i de refererte attributtene
- Dette må sjekkes av systemet både ved **insert**, **update** og **delete**

drop, alter

drop table R;

alter table R **add** A_x D_y;

alter table R **drop** A_x;

- R er et relasjonsnavn
- A_x er et attributt
- D_y er et domene

insert (fra SQLs DML)

- **insert**: Innsetting av nye data

insert into $R(A_1, A_2, \ldots, A_k)$
values $(v_1, v_2, \ldots, v_k)$;

- Attributtlisten kan sløyfes hvis den dekker samtlige attributter i R og følger attributtenes default rekkefølge

Mer om SQLs DML, se lenger ned.

SQLs DQL

select [**distinct**] ATTRIBUTTLISTE
from NAVNELISTE
[**where** WHERE-BETINGELSE]
[**group by** GRUPPERINGSATTRIBUTTER
[**having** HAVING-BETINGELSE]]
[**order by** ATTRIBUTT [**asc** | **desc**]
[, ATTRIBUTT [**asc** | **desc**]] ...];

[ ] betyr at dette leddet er en valgfri del av setningen

Select-setningens enkeltdeler

- **select**
  Angir hvilke attributter som skal vises i svaret
- **distinct**
  Fjerner flerforekomster (duplikater) av svartuplene
- **from**
  Navn på de relasjonene spørringen refererer til
- **where**
  Seleksjonsbetingelse (kan inneholde en eller flere join-betingelser)
- **group by**, **having**
  Angir grupperingsattributter til bruk ved aggregering og betingelser på resultatet av grupperingen; disse kommer vi tilbake til i en senere forelesning
- **order by** Ordner tuplene i henhold til angitte kriterier

Select-setningen

Typisk utseende:

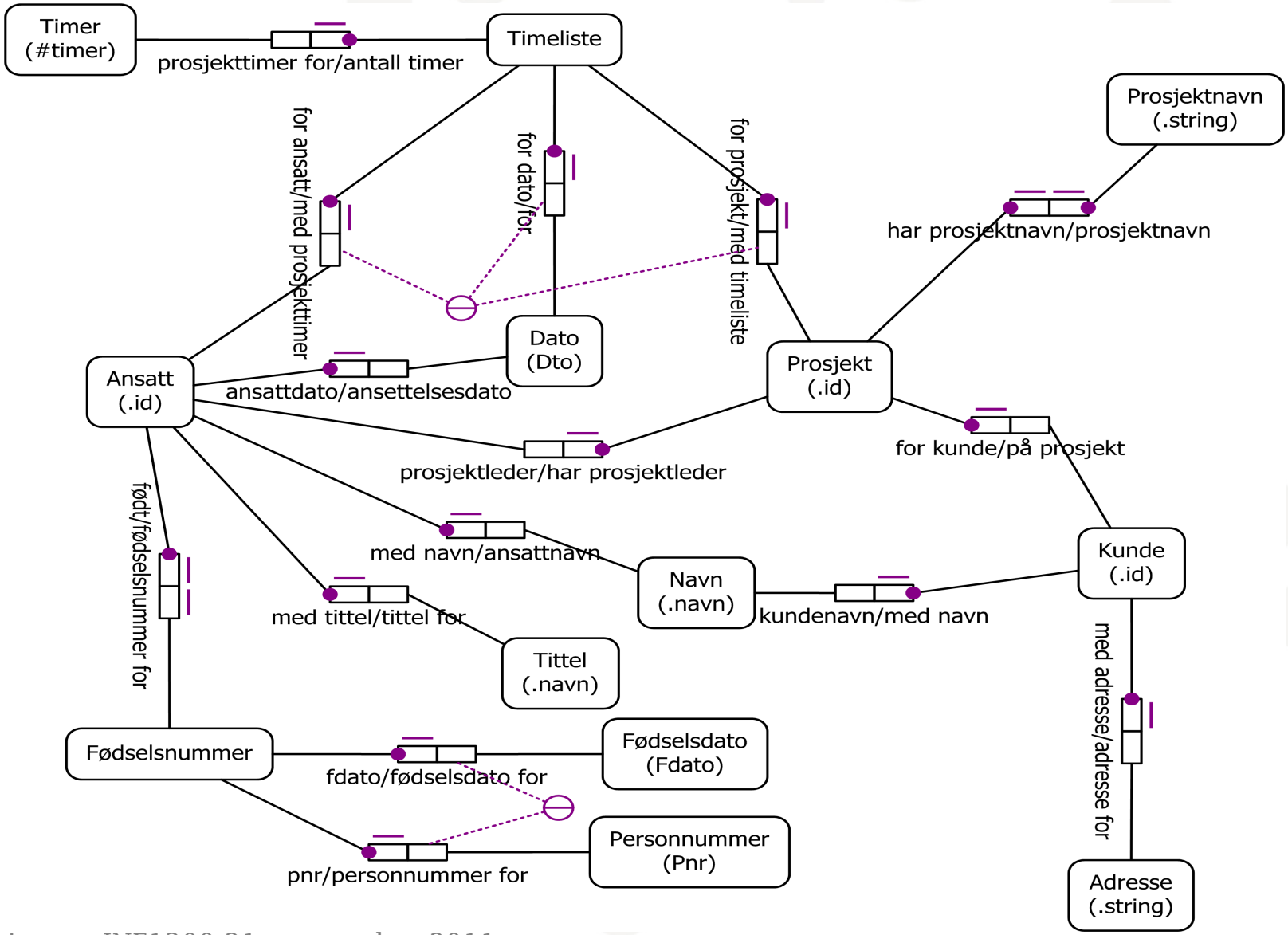
select [**distinct**] $A_1, A_2, \ldots, A_j$
from $R_1, R_2, \ldots, R_k$
where C;

hvor

$R_1, R_2, \ldots, R_k$ er relasjonsnavn

$A_1, \ldots, A_j$ er attributter fra $R_1, R_2, \ldots, R_k$

C er en betingelse


Timer
(#timer)
prosjekttimer for/antall timer
Timeliste
for ansatt/med prosjekttimer
for dato/for
for prosjekt/med timeliste
Prosjektnavn
(.string)
har prosjektnavn/prosjektnavn
Dato
(Dto)
Ansatt
(.id)
ansattdato/ansettelsesdato
Prosjekt
(.id)
for kunde/på prosjekt
prosjektleder/har prosjektleder
født/fødselsnummer for
med navn/ansattnavn
Kunde
(.id)
Navn
(.navn)
kundenavn/med navn
med tittel/tittel for
Tittel
(.navn)
med adresse/adresse for
Fødselsnummer
Fødselsdato
(Fdato)
fdato/fødselsdato for
Personnummer
(Pnr)
pnr/personnummer for
Adresse
(.string)

select—eksempel 1

Skjema

Prosjekt(PId, Pnavn, KId, Pleder, StartDato)
Ansatt(AId, Navn, Tittel, Fdato, Pnr, AnsDato)
Timeliste(AId, Dato, PId, Timer)
Kunde(KId, Knavn, Adresse)

- **Oppgave:** Finn navn på de ansatte som er ansatt etter 2003. (Det kan være flere ansatte som har samme navn.)
- **Løsning**

```
select distinct Navn
from Ansatt
where AnsDato > date '2003-12-31'
```

Merknader til *select*

- **select** (SQL) skiller ikke mellom store og små bokstaver, unntatt i tekststrenger
- **select** beregner *bager* (med unntak av noen av operatorene)

En *bag* er en *mengde* med tupler der samme tuppel kan forekomme flere ganger. (Like tupler fjernes eventuelt ved å bruke **distinct**.)

select—eksempel 2

Skjema

Prosjekt(PId, Pnavn, KId, Pleder, StartDato)
Ansatt(AId, Navn, Tittel, Fdato, Pnr, AnsDato)
Timeliste(AId, Dato, PId, Timer)
Kunde(KId, Knavn, Adresse)

- **Oppgave**
  Finn navn og startdato for alle prosjekter bestilt av kunden «Pust og pes AS». Sorter dem slik at det nyeste prosjektet kommer først.

- **Løsning**

```
select   Pnavn, StartDato
from     Kunde K, Prosjekt P
where    Knavn ='Pust og pes AS' and K.KId = P.KId
order by StartDato desc;
```

Seleksjons- og join-betingelser

La oss se nærmere på løsningen fra forrige lysark:

```
select   Pnavn, StartDato
from     Kunde K, Prosjekt P
where    Knavn ='Pust og pes AS' and K.KId = P.KId
order by StartDato desc;
```

where-betingelsen består av to deler:

- Knavn = 'Pust og pes AS'
  Dette leddet kalles en **seleksjonsbetingelse**
  Det plukker ut forekomster i Kunde (her trolig bare en)
- K.KId = P.Kid
  Dette leddet kalles en **join-betingelse**
  Det kobler sammen forekomster fra Kunde med forekomster i Prosjekt forutsatt at verdiene i attributtene KId og Kid er like

Uttrykk i betingelser — 1

where-betingelsen er et boolsk uttrykk hvor atomene har en av følgende former:

- Verdisammenlikning: P **op** Q
  - P og Q må ha samme domene, minst en av dem må være et attributt, den andre kan være en konstant
  - **op** $\in$ {=, <, >, <=, >=, <>, **like**}
    (**like** er bare lov når Q er en konstant tekststreng)
- null-test: P **is null** eller P **is not null**
- Relasjonssammenlikning: **exists**, **in**, **all**, **any**
  (Disse tar vi for oss i en senere forelesning)

Uttrykk i betingelser — 2

Spesialregler for sammenlikning av **strenger** :

- Leksikografisk ordning: $s < t$, $s > t$, $s <= t$, $s >= t$
- Sammenlikning: $s = t$, $s <> t$
- Mønstergjenkjenning: s **like** p
  p er et mønster hvor
  % matcher en vilkårlig sekvens (null eller flere tegn)
  _ matcher ett vilkårlig tegn

Uttrykk i betingelser — 3

Datoer og tidspunkter:

- Dato: **date** 'yyyy-mm-dd'
- Tidspunkt: **time** 'hh:mm', **time** 'hh:mm:ss'
- Tidspunkt med finere gradering enn sekund: **time** 'hh:mm:ss.ccc'
- Tidspunkt før GMT: **time** 'hh:mm:ss+hh'
- Tidspunkt etter GMT: **time** 'hh:mm:ss-hh'
- Dato og tid: **timestamp** 'yyyy-mm-dd hh:mm:ss'

select—eksempel 3

Skjema
Prosjekt(PId, Pnavn, KId, Pleder, StartDato)
Ansatt(AId, Navn, Tittel, Fdato, Pnr, AnsDato)
Timeliste(AId, Dato, PId, Timer)
Kunde(KId, Knavn, Adresse)

- *Oppgave*: Finn navn og tittel på alle som har arbeidet på prosjektet «Vintersalg»
- *Løsning*

```
select distinct Navn, Tittel
from    Ansatt A, Timeliste T, Prosjekt P
where   Pnavn = 'Vintersalg' and
        P.PId = T.PId and T.AId = A.AId;
```

- Her består join-betingelsen av to ledd. Den binder sammen en forekomst fra hver av de tre tabellene Ansatt, Timeliste og Prosjekt
- At join-attributtene parvis har samme navn, er tilfeldig. Det holder at de har samme domene

select—navnekonflikter

- Kvalifiser attributter med relasjonsnavn: R.A
- Navngi relasjoner med aliaser:
  ...**from** R **as** S... (**as** kan sløyfes)
  S blir en kopi av R med nytt relasjonsnavn
- Gi attributter nytt navn:
  select A **as** B **from**...
  A omnavnes til B i resultatrelasjonen

SQLs DML

- **insert**: Innsetting av nye data
- **update**: Endring av eksisterende data
- **delete**: Sletting av data

insert

insert into $R(A_1, A_2, \ldots, A_k)$
values $(v_1, v_2, \ldots, v_k)$;

insert into $R(A_1, A_2, \ldots, A_k)$
select-setning;

- Attributtlisten kan sløyfes hvis den dekker samtlige attributter i R og følger attributtenes default rekkefølge
- **NB**—optimaliseringer i DBMSet kan medføre at tuplene legges inn etterhvert som de beregnes i **select**-setningen. Dette kan ha sideeffekter på beregningen av **select**-setningen

update, delete

update R
set $A_1 = E_1, \ldots, A_k = E_k$
[**where** C];

delete from R
[**where** C];

R er en relasjon, A_i er attributter (kolonnenavn) og E_j uttrykk. [ ] betyr at dette leddet er en valgfri del av setningen

SQLs SDL: indekser

create index X on $R(A_1, \ldots, A_k)$;

drop index X;

Valg av indekser må gjøres med omhu.
Indekser gjør at

- spørringer mot vedkommende attributt(er) går mye fortere
- innsetting, sletting og oppdatering blir mer komplisert

Implementasjon av indekser

- Implementasjon av indekser:
  - B+-trær
  - Hash
  - ...
- For mer informasjon, se INF2220—Algoritmer og datastrukturer

Indekser på kandidatnøkler

- DBMSer bygger vanligvis indekser automatisk på primærnøklene
- For hver kandidatnøkkel må man vurdere spesielt om det bør deklareres indeks på nøkkelen. Syntaks avhenger av DBMSet Noen SQL-implementasjoner tillater deklarasjon av kandidatnøkkel + indeks i en og samme setning:

```
create unique index FnrIndex
on Ansatt (Fdato, Pnr);
```

- **I Postgres bygges automatisk en unique index på kandidatnøkler**
- Hvis det er opprettet indeks på en nøkkel, benyttes denne under sjekk av flerforekomster. Ellers: Må i verste fall søke gjennom hele relasjonen

PostgreSQL

- For å aksessere filmdatabasen: Fra Linux-promptet (**...>**), gi kommandoen
  > psql -h kurspg -U <brukernavn> -d fdb
  og du blir bedt om å oppgi PostgreSQL-passordet ditt. kurspg er vertsmaskinen, <brukernavn> er ditt Postgres-brukernavn, og fdb er navnet på filmdatabasen
- Dersom du vil lage egne tabeller, skriver du ditt eget brukernavn i stedet for fdb
- For å kjøre en kommandofil, skriv \i <filnavn>
- For å avslutte, skriv \q
- Les forøvrig dokumentet om filmdatabasen og postgres som er tilgjengelig via lenke fra kursets semesterside.