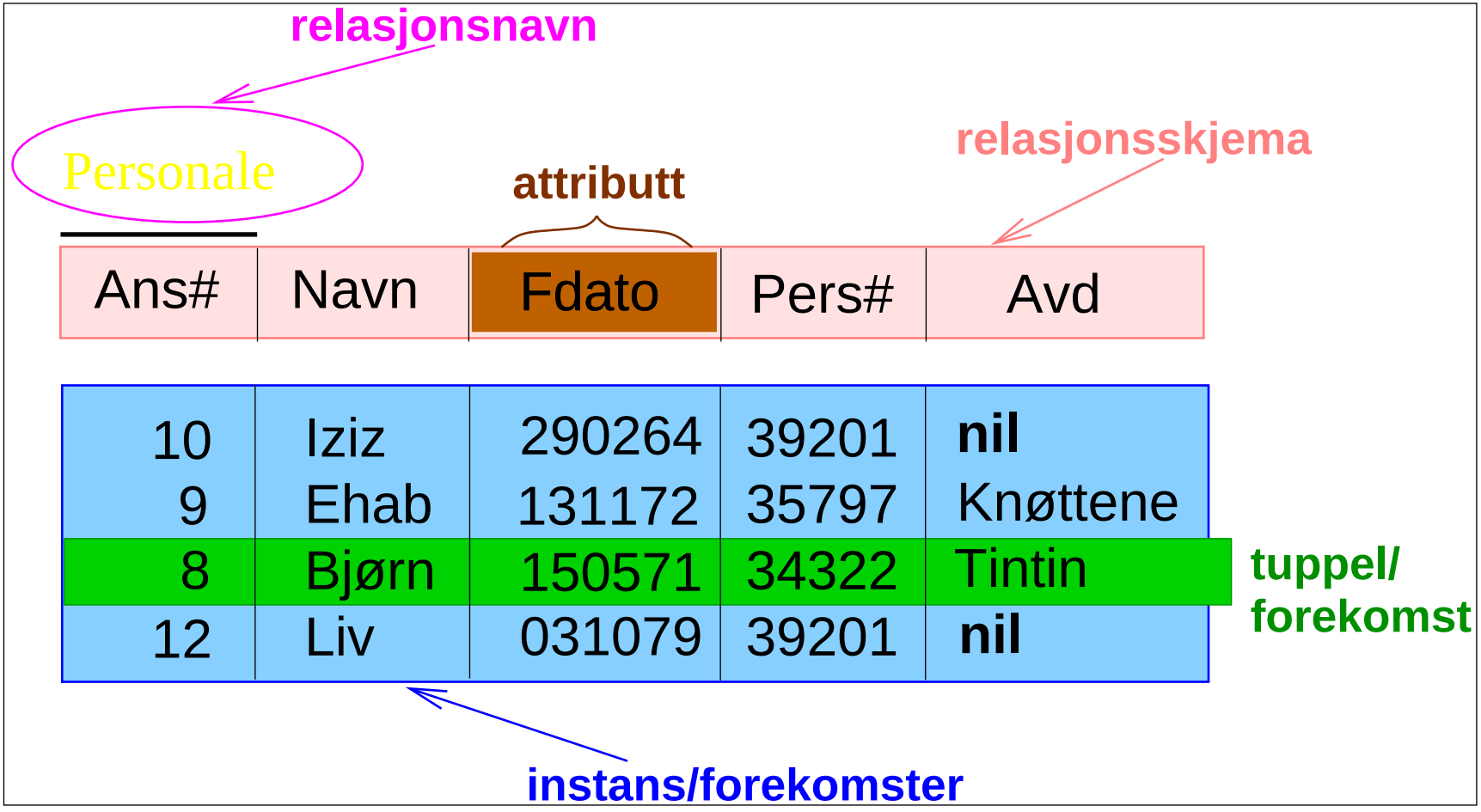
- Litt tabellterminologi
- Definere tabeller
- Fylle tabeller med data
- Hente data fra tabeller
  - select-from-where
  - distinct
  - order by

Relasjoner—terminologi



tabell = relasjon

Lage en tabell med SQL—forenklet uten skranker

create table R (
A_1 D_1,
A_2 D_2,
...
A_n D_n
);

R er navnet på relasjonen/tabellen
A_i er et atributt
D_j er et domene

create—eksempel

```
create table EmneStudent (
        brukernavn varchar(8),
        emnekode varchar(7),
        gruppenr integer,
        eksamensdato date
);
```

create—eksempel

```
create table EmneStudent (
        brukernavn varchar(8),
        emnekode varchar(7),
        gruppenr int, -- kan også skrive integer
        eksamensdato date
);
```

Legg merke til likheten med en klassedefinisjon i Java:

```
public class EmneStudent {
        String brukernavn;
        String emnekode;
        int gruppenr;
        Date eksamensdato;
}
```

Datatyper i PostgreSQL

datatype	*forklaring*
`integer`	heltall
`real`	flyttall
`numeric(n,d)`	*n* desimale sifre, *d* etter desimalpunktum
`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

Datatyper sortert etter hyppighet i INF1300

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

create—nytt eksempel

Student(bnavn, snr, navn, stprog)

```
create table Student (
        bnavn char(8) primary key,
        snr int unique,
        navn varchar(80) not null,
        stprog varchar(40)
);
```

create—nytt eksempel

Student(bnavn, snr, navn, stprog)

- bnavn er studentens brukernavn

```
create table Student (
        bnavn char(8) primary key,
        snr int unique,
        navn varchar(80) not null,
        stprog varchar(40)
);
```

create—nytt eksempel

Student(bnavn, snr, navn, stprog)

- bnavn er studentens brukernavn
- snr er studentens studentnummer

```
create table Student (
        bnavn char(8) primary key,
        snr int unique,
        navn varchar(80) not null,
        stprog varchar(40)
);
```

create—nytt eksempel

Student(bnavn, snr, navn, stprog)

- bnavn er studentens brukernavn
- snr er studentens studentnummer
- stprog er studieprogrammet studenten er tatt opp til

```
create table Student (
        bnavn char(8) primary key,
        snr int unique,
        navn varchar(80) not null,
        stprog varchar(40)
);
```

Legge inn data i tabeller

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

Legge inn data i tabeller

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

```
insert into Student (snr, navn, bnavn, stprog)
values (133423, 'Liv E. Laga', 'livelaga',
        'Informatikk: programmering og nettverk'
);

insert into Student
values ('alistrak', 133424, 'Ali Straks',
       'Informatikk: design, bruk, interaksjon'
);
```

Husk attributtrekkefølgen fra definisjonen av Student:
Student (bnavn, snr, navn, stprog)

Hente/skrive ut data fra tabeller

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, så i dag skal vi bare se på denne delen:

Hente/skrive ut data fra tabeller

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

[ ] betyr at dette leddet er en valgfri del av setningen, så i sin enkleste form ser setningen slik ut:

Hente/skrive ut data fra tabeller

select ATTRIBUTTLISTE
from NAVNELISTE
;

Eksempel:

```
select bnavn, snr, navn, stprog
from Student
;
```

samme som:

```
select * from Student ;
```

Student (bnavn, snr, navn, stprog)

Hente/skrive ut data fra tabeller

select ATTRIBUTTLISTE
from NAVNELISTE
;

Noen attributter:

```
select bnavn, navn
from Student
;
```

annen rekkefølge:

```
select snr, navn, bnavn, stprog
from Student ;
```

Student (bnavn, snr, navn, stprog)

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)
- **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

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.)

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 *samling* med tupler der samme tuppel kan forekomme flere ganger. (Til forskjell fra en *mengde* tupler/forekomster, der samme tuppel ikke 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.

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

Tekstmønstre

- I SQL kan vi bruke **like** for å sammenligne et tekst-attributt med et tekstmønster

Eksempel 1:

```
select firstname from person
where firstname like 'O_a';
```

passer med Oda og Ola og O4a, men *ikke* med Olga

Eksempel 2:

```
select firstname from person
where firstname like 'O%a';
```

passer med alle navn som begynner med «O» og slutter med «a», som Ola, Olga, Othilia, Oda, Ofjhwskjfhkxxa

Tekstmønstre

- I SQL kan vi bruke **like** for å sammenligne et tekst-attributt med et tekstmønster
- Et *tekstmønster* er en tekstkonstant hvor to tegn, kalt jokertegn, har spesiell betydning:
  - _ (understrekning) passer med *ett* vilkårlig tegn
  - % passer med en vilkårlig tekststreng (null eller flere tegn)

Eksempel 1:

```
select firstname from person
where firstname like 'O_a';
```

passer med Oda og Ola og O4a, men *ikke* med Olga

Eksempel 2:

```
select firstname from person
where firstname like 'O%a';
```

passer med alle navn som begynner med «O» og slutter med «a», som Ola, Olga, Othilia, Oda, Ofjhwskjfhkxxa

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»

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;
```

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

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

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

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

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;

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

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

PostgreSQL

- For å bruke PostgreSQL: Fra Linux-promptet (**...>**), gi kommandoen

  ```
  > psql -h dbpg-ifi-kurs -U <brukernavn>
  ```

  og du blir bedt om å oppgi passordet ditt.
- 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.