- Mer SQL:
- kandidat-, primær- og fremmednøkler
- Definere tabeller med integritetsregler
- Hente data fra tabeller
  - select-from-where
  - distinct
  - order by
- Eksempler kjøres i PostgreSQL (psql)

Lage en tabell med SQL

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

R er navnet på relasjonen/tabellen
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

Emner(emnekode, emnenavn)
StudTarEmne(bnavn, emne)

```
create table StudTarEmne (
        bnavn char(8),
        emne varchar(12)
);

create table Emner (
        emnekode varchar(12) primary key,
        emnenavn varchar(80),
        semester char(6) check (semester='spring'
                                or semester='fall')
);
```

Nøkler og nøkkelattributter

Nøklene i en relasjon er de viktigste skrankene. En kandidatnøkkel er kort sagt et eller flere attributter som bare kan ha unike verdier. F.eks. (fødselsdato, personnr), eller (brukernavn), eller (filmid), eller (brukernavn, emnekode, semester).

Kandidatnøkkel:

En minimal kombinasjon (minimal delmengde) X av attributtene $\{A_1, A_2, \ldots, A_n\}$ som er slik at hvis t og u er to tupler hvor $t \neq u$, så er $t[X] \neq u[X]$.

En minimal mengde attributter i en tabell som entydig identifiserer ethvert tuppel i tabellen.

Primærnøkkel: En utvalgt blant kandidatnøklene. I relasjonsmodellen skal alle relasjoner ha nøyaktig én primærnøkkel. Primærnøkkelen blir markert med én strek.

Hvis det er andre kandidatnøkler, markeres de med én strek, og primærnøkkelen med to[1].

Nøkkelattributt: Attributt som er med i (minst) en kandidatnøkkel.

[1] Vi har i INF1300 gjort det omvendt tidligere, men har endret dette fra 2015 for å bli konsistent med læreboka.

Primærnøkler

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

```
create table Emner (
        emnekode varchar(12) primary key,
        ...
);
```

Primærnøkler

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

```
create table Emner (
        emnekode varchar(12) primary key,
        ...
);
```

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

```
create table Emner (
        emnekode varchar(12),
        ...
        primary key(emnekode)
);
```

Primærnøkler

- Hvis primærnøkkelen består av flere attributter må definisjonen kome etter attributtdeklarasjonene

```
create table eksamensforsøk (
        brukernavn char(8)
        emne   char(10)
        semester char(5),
        karakter char(1),
        primary key (brukernavn, emne, semester)
);
```

Forsøk på å legge inn flere karakterer for samme student i samme emne i samme semester vil da feile.

Primærnøkler, regler

- Maks én primærnøkkeldeklarasjon pr. relasjon

Primærnøkler, regler

- Maks én primærnøkkeldeklarasjon pr. relasjon
- Flere kandidatnøkler, bruk skranken **unique**

Primærnøkler, regler

- Maks én primærnøkkeldeklarasjon pr. relasjon
- Flere kandidatnøkler, bruk skranken **unique**
- 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**

Primærnøkler, regler

- Maks én primærnøkkeldeklarasjon pr. relasjon
- Flere kandidatnøkler, bruk skranken **unique**
- 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**

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

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='Konsulent' or ...
              ),
        ...
    );
    ```
  - Angir en betingelse på attributtet. Sjekkes ved hver endring av attributtets verdi
    ```

Skranke på ett attributt

- check
  - ```
    create table Ansatt (
        Navn varchar(40),
        Tittel varchar(15),
        Fnr varchar(11),
        ... ,
        CONSTRAINT sjekkTittel check
            ( Tittel='Selger' or
              Tittel='Konsulent' or ...),
        ...
    );
    ```
    ```

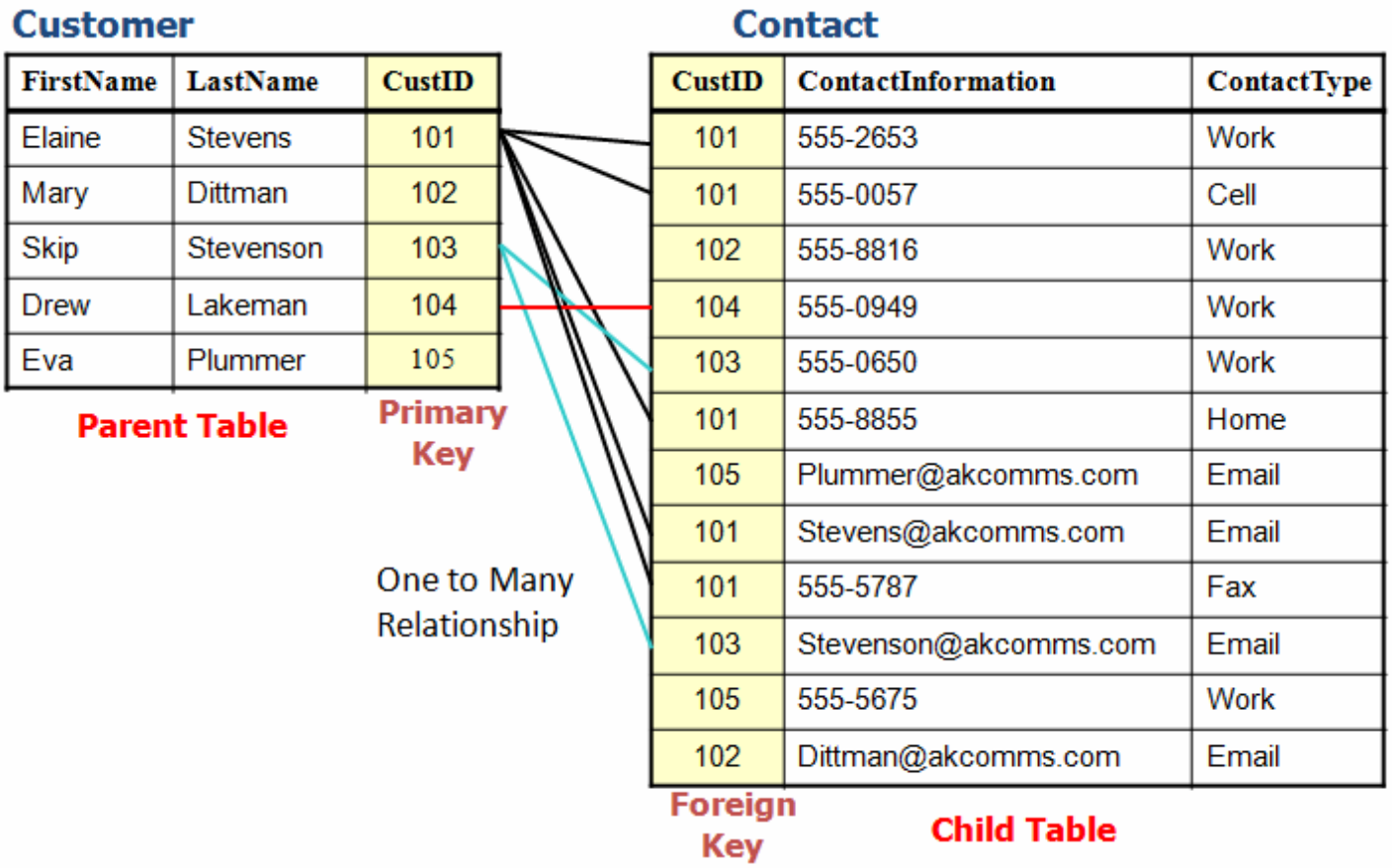
Skranke på ett attributt

- check
  - ```
    create table Ansatt (
        Navn varchar(40),
        Tittel varchar(15),
        Fnr varchar(11),
        ...,
        CONSTRAINT sjekkTittel check
            ( Tittel='Selger' or
              Tittel='Konsulent' or ...),
        ...
    );
    ```
- check
  - ```
    create table Emne (
        eKode varchar(7),
        eNavn varchar(99),
        ...,
        CONSTRAINT sjekkEkode check (ekode > 'INF0000'
                                     and ekode < 'INF9999'),
        ...
    );
    ```

Skranke på ett attributt

- check
  - ```
    create table Ansatt (
        Navn varchar(40),
        Tittel varchar(15),
        Fnr varchar(11),
        ... ,
        CONSTRAINT sjekkTittel check
            ( Tittel='Selger' or
              Tittel='Konsulent' or ...),
        ...
    );
    ```
- check
  - ```
    create table Emne (
        eKode varchar(7),
        eNavn varchar(99),
        ... ,
        CONSTRAINT sjekkEkode
                    check (ekode like 'INF____'),
        ...
    );
    ```

Skranker: fremmednøkler



Hva skal vi med fremmednøkler?

- Hva skal vi med fremmednøkler?

- Hva skal vi med fremmednøkler?
- Knytte sammen informasjon som hører sammen (f.eks. til samme objekt i virkeligheten) fra flere tabeller.

- Hva skal vi med fremmednøkler?
- Knytte sammen informasjon som hører sammen (f.eks. til samme objekt i virkeligheten) fra flere tabeller.
- Må vi definere fremmednøkler for å gjøre dette?

- Hva skal vi med fremmednøkler?
- Knytte sammen informasjon som hører sammen (f.eks. til samme objekt i virkeligheten) fra flere tabeller.
- Må vi definere fremmednøkler for å gjøre dette?
- Nei

- Hva skal vi med fremmednøkler?
- Knytte sammen informasjon som hører sammen (f.eks. til samme objekt i virkeligheten) fra flere tabeller.
- Må vi definere fremmednøkler for å gjøre dette?
- Nei
- Hvorfor gjør vi det da?

- Hva skal vi med fremmednøkler?
- Knytte sammen informasjon som hører sammen (f.eks. til samme objekt i virkeligheten) fra flere tabeller.
- Må vi definere fremmednøkler for å gjøre dette?
- Nei
- Hvorfor gjør vi det da?
- For å sikre at det ikke er mulig å blande sammen informasjon som ikke hører sammen i virkeligheten (i UoD).

- Hva skal vi med fremmednøkler?
- Knytte sammen informasjon som hører sammen (f.eks. til samme objekt i virkeligheten) fra flere tabeller.
- Må vi definere fremmednøkler for å gjøre dette?
- Nei
- Hvorfor gjør vi det da?
- For å sikre at det ikke er mulig å blande sammen informasjon som ikke hører sammen i virkeligheten (i UoD).
- F.eks. at en person har ett navn, ikke flere, eller ingen.

- Hva skal vi med fremmednøkler?
- Knytte sammen informasjon som hører sammen (f.eks. til samme objekt i virkeligheten) fra flere tabeller.
- Må vi definere fremmednøkler for å gjøre dette?
- Nei
- Hvorfor gjør vi det da?
- For å sikre at det ikke er mulig å blande sammen informasjon som ikke hører sammen i virkeligheten (i UoD).
- F.eks. at en person har ett navn, ikke flere, eller ingen.
- For at DBMSet skal gjøre oppslag i tabeller mer effektivt.

Skranker: fremmednøkler

- *med ett attributt, samme attributtnavn*:

```
create table StudTarEmne (
    ....  ,
    emnekode char(10) references Emner  ,
    ....
);
```

- *med ett attributt, ulike attributtnavn*:

```
create table StudTarEmne (
    ....  ,
    emnekode char(10) references Kurs(kurskode)   ,
    ....
);
```

- *med flere attributter med samme navn*:

```
create table StudFikkKarakterIEmne (
    studid int ,
    emnekode varchar(10) ,
    semester varchar(5) ,
    karakter char(1) ,

    foreign key (studid, emnekode, semester)
            references Resultater

);
```

De refererte attributtene må være primærnøkkel, deklarert **primary key**. (I noen SQL-systemer holder det med at attributtene er deklarert UNIQUE, men ikke i PostgreSQL).

- *med flere attributter med forskjellig navn*:

```
create table StudFikkKarakterIEmne (
  studid int ,
  emnekode varchar(10) ,
  semester varchar(5) ,
  karakter char(1) ,

  foreign key (studid, emnekode, semester)
          references Resultater (bnavn, kurskode, sem)

);
```

De refererte attributtene må være primærnøkkel, deklarert **primary key**. (I noen SQL-systemer holder det med at attributtene er deklarert UNIQUE, men ikke i PostgreSQL).

Fremmednøkler mot flere tabeller brukes for å implementere et mange-til-mange forhold mellom tabeller:

```
create table student (
    bnavn char(8) primary key,
    navn varchar(80),
    ...
);

create table emne (
    ekode char(10) primary key,
    emnenavn varchar(80),
    emneeier varchar(80),
    ...
);

create table antalleksamensforsøk (
    brukernavn char(8) references student(bnavn),
    emne  char(10) references emne,
    antforsøk integer,
    primary key (brukernavn, emne)
);
```

Tilbake til ...

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:

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)
- AND, OR, NOT og ( ) som vanlig i boolske uttrykk
- 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'

Tekstmønstre

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

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)

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

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

SELECT beregner *bager*

- **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. (Like tupler fjernes eventuelt ved å bruke **distinct**.)

select—eksempel

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

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

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

select—eksempel

Skjema

Student(bnavn, navn, uid, spkode, grlremne, grlrgrnr)
StudentTarEmne(bnavn, emne)
Emne(emnekode, enavn, fsem)

- **Oppgave**
  Skriv ut studentnavn eg emnenavn og på alle studenter som har brukernavn som begynner med bokstaven 'a' og som tar emner på 1000-nivå. Sorter dem etter brukernavn.
- **Løsning**

```
select S.navn, E.enavn
from Student S, Emne E, StudTarEmne STE
where
    -- seleksjonsbetingelse:
    ( S.bnavn like 'a%' and STE.emne like 'inf1%' )

    -- joinbetingelse:
    and  (     S.bnavn = STE.bnavn
           and E.emnekode = STE.emne )
order by S.bnavn;
```

PostgreSQL

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

  ```
  > psql -h dbpg-ifi-kurs -U <brukernavn> [-d fdb]
  ```

  og du blir bedt om å oppgi passordet ditt.
- Hvis du vil koble deg opp ot filmdatabasen tar du med `-d fdb` på slutten av linja
- 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.

```sql
select * from emne
select * from studtaremne
select distinct emnekode, fsem from emne;
select emnekode, fsem from emne;
select fsem from emne;
...
select *
from emne, studtaremne
where emne = 'inf1300' and emne = emnekode;

select S.navn, E.enavn
from Student S, Emne E, StudTarEmne STE
where
  -- seleksjonsbetingelse:
  ( S.bnavn like 'a%' and STE.emne like 'inf1%' )
  -- joinbetingelse:
  and  (    S.bnavn = STE.bnavn
        and E.emnekode = STE.emne )
order by S.bnavn;

select * from film where title like 'Garc';

select * from film f, person p where
f.title = p.firstname and f.title like 'Ac%';

select * from film f, person p where
f.title = p.firstname and f.prodyear = 1931;

select * from InnbILand A, InnbILand B
where A.navn <> B.navn and A.innb = B.innb;
```