NULL verdier

- Alle attributter har NULL som mulig verdi
  - mulige verdier for integer: NULL, 0, 1, 2, 3...
- Dog mulig å lage tabeller med attributter som *forbyr* NULL
- Ulik bruk:
  - manglende informasjon («vet ikke hvor mye lønn Kari får»)
  - uaktuell informasjon («Kari får ikke lønn»)
  - (av og til også annet)
- Mulig å utforme databaser slik at de ikke trenger NULL
- Forekommer ofte i praksis

Spørringer med NULL

P			
ID	**navn**	**jobb**	**lønn**
1	Kari	manager	800000
2	Petter	kokk	500000
3	Ola	NULL	450000
4	Johnny	NULL	NULL

- `SELECT navn FROM P WHERE lønn<600000`

Spørringer med NULL

P			
ID	**navn**	**jobb**	**lønn**
1	Kari	manager	800000
2	Petter	kokk	500000
3	Ola	NULL	450000
4	Johnny	NULL	NULL

- `SELECT navn FROM P WHERE lønn<600000`
  - Petter, Ola

Spørringer med NULL

P			
ID	**navn**	**jobb**	**lønn**
1	Kari	manager	800000
2	Petter	kokk	500000
3	Ola	NULL	450000
4	Johnny	NULL	NULL

- `SELECT navn FROM P WHERE lønn<600000`
  - Petter, Ola
- `SELECT navn FROM P WHERE not lønn<600000`

Spørringer med NULL

P			
ID	**navn**	**jobb**	**lønn**
1	Kari	manager	800000
2	Petter	kokk	500000
3	Ola	NULL	450000
4	Johnny	NULL	NULL

- `SELECT navn FROM P WHERE lønn<600000`
  - Petter, Ola
- `SELECT navn FROM P WHERE not lønn<600000`
  - Kari

Spørringer med NULL

P			
ID	**navn**	**jobb**	**lønn**
1	Kari	manager	800000
2	Petter	kokk	500000
3	Ola	NULL	450000
4	Johnny	NULL	NULL

- `SELECT navn FROM P WHERE lønn<600000`
  - Petter, Ola
- `SELECT navn FROM P WHERE not lønn<600000`
  - Kari
- Sammenligning mellom NULL for Johnny og 600000 gir «UNKNOWN»

Spørringer med NULL

P			
ID	**navn**	**jobb**	**lønn**
1	Kari	manager	800000
2	Petter	kokk	500000
3	Ola	NULL	450000
4	Johnny	NULL	NULL

- `SELECT navn FROM P WHERE lønn<600000`
  - Petter, Ola
- `SELECT navn FROM P WHERE not lønn<600000`
  - Kari
- Sammenligning mellom NULL for Johnny og 600000 gir «UNKNOWN»
- SELECT viser tupler hvor WHERE er TRUE, ikke FALSE eller UNKNOWN

Spørringer med NULL

P			
ID	**navn**	**jobb**	**lønn**
1	Kari	manager	800000
2	Petter	kokk	500000
3	Ola	NULL	450000
4	Johnny	NULL	NULL

- `SELECT navn FROM P WHERE lønn = NULL`

Spørringer med NULL

P			
ID	**navn**	**jobb**	**lønn**
1	Kari	manager	800000
2	Petter	kokk	500000
3	Ola	NULL	450000
4	Johnny	NULL	NULL

- `SELECT navn FROM P WHERE lønn = NULL`
  - ingen resultater... ingenting er «lik» NULL

Spørringer med NULL

P			
ID	**navn**	**jobb**	**lønn**
1	Kari	manager	800000
2	Petter	kokk	500000
3	Ola	NULL	450000
4	Johnny	NULL	NULL

- `SELECT navn FROM P WHERE lønn = NULL`
  - ingen resultater... ingenting er «lik» NULL
- `SELECT navn FROM P WHERE lønn IS NULL`

Spørringer med NULL

P			
ID	**navn**	**jobb**	**lønn**
1	Kari	manager	800000
2	Petter	kokk	500000
3	Ola	NULL	450000
4	Johnny	NULL	NULL

- `SELECT navn FROM P WHERE lønn = NULL`
  - ingen resultater... ingenting er «lik» NULL
- `SELECT navn FROM P WHERE lønn IS NULL`
  - Johnny

Spørringer med NULL

P			
ID	**navn**	**jobb**	**lønn**
1	Kari	manager	800000
2	Petter	kokk	500000
3	Ola	NULL	450000
4	Johnny	NULL	NULL

- `SELECT navn FROM P WHERE lønn = NULL`
  - ingen resultater... ingenting er «lik» NULL
- `SELECT navn FROM P WHERE lønn IS NULL`
  - Johnny
- `SELECT navn, jobb FROM P WHERE lønn IS NOT NULL`

Spørringer med NULL

P			
ID	**navn**	**jobb**	**lønn**
1	Kari	manager	800000
2	Petter	kokk	500000
3	Ola	NULL	450000
4	Johnny	NULL	NULL

- `SELECT navn FROM P WHERE lønn = NULL`
  - ingen resultater... ingenting er «lik» NULL
- `SELECT navn FROM P WHERE lønn IS NULL`
  - Johnny
- `SELECT navn, jobb FROM P WHERE lønn IS NOT NULL`
  -

navn	**jobb**
Kari	manager
Petter	kokk
Ola	

Spørringer med NULL

P			
ID	**navn**	**jobb**	**lønn**
1	Kari	manager	800000
2	Petter	kokk	500000
3	Ola	NULL	450000
4	Johnny	NULL	NULL

- `SELECT navn FROM P WHERE lønn = NULL`
  - ingen resultater... ingenting er «lik» NULL
- `SELECT navn FROM P WHERE lønn IS NULL`
  - Johnny
- `SELECT navn, jobb FROM P WHERE lønn IS NOT NULL`
  -

navn	jobb
Kari	manager
Petter	kokk
Ola	

- NULL vises i resultater som tom celle.

Spørringer som regner

- Liste opp datoer og tiemr jobbet:
  `select dato, timer from timeliste ;`

Spørringer som regner

- Liste opp datoer og tiemr jobbet:
  `select dato, timer from timeliste ;`
- Hva hvis vi trenger minutter?

Spørringer som regner

- Liste opp datoer og tiemr jobbet:
  `select dato, timer from timeliste ;`
- Hva hvis vi trenger minutter?
- Mulig å regne i select delen:
  `select dato, timer*60 as minutter from timeliste ;`

Spørringer som regner

- Liste opp datoer og tiemr jobbet:
  `select dato, timer from timeliste ;`
- Hva hvis vi trenger minutter?
- Mulig å regne i select delen:
  `select dato, timer*60 as minutter from timeliste ;`
- Kan godt kombinere flere attributter i en formel
  `select ...sqrt(x*x+y*y) as d...`

Spørringer som regner

- Liste opp datoer og tiemr jobbet:
  `select dato, timer from timeliste ;`
- Hva hvis vi trenger minutter?
- Mulig å regne i select delen:
  `select dato, timer*60 as minutter from timeliste ;`
- Kan godt kombinere flere attributter i en formel
  `select ...sqrt(x*x+y*y) as d...`

- Men hva hvis vi vil telle sammen alle timer på samme dato?

Aggregeringsfunksjoner

SQL har forskjellige aggregeringsfunksjoner.
De 5 vanligste:

navn	*virkning (returnerer)*
count	teller antall
min	finner minste verdi
max	finner største verdi
sum	summerer verdier
avg	finner gjennomsnitt av verdier

PostgreSQL: `https://www.postgresql.org/docs/9.6/static/functions-aggregate.html`

ANSI SQL: `https://www.safaribooksonline.com/library/view/sql-in-a/9780596155322/ch04s02.html`

count()

- `select count(*) from person;`

  gir antall tupler i tabellen

count()

- ```
  select count(*) from person;
  ```

  gir antall tupler i tabellen

- ```
  select count(*) as antTupler from person;
  ```

  Som for alle attributter i select-listen, kan vi gi **count**(*) et nytt navn.

count()

- `select count(*) from person;`

  gir antall tupler i tabellen

- `select count(*) as antTupler from person;`

  Som for alle attributter i select-listen, kan vi gi **count**(*) et nytt navn.

- `select count(gender) from person;`

  gir antall tupler i tabellen hvor attributtet gender ikke er null

count()

- `select count(*) from person;`

  gir antall tupler i tabellen

- `select count(*) as antTupler from person;`

  Som for alle attributter i select-listen, kan vi gi **count**(*) et nytt navn.

- `select count(gender) from person;`

  gir antall tupler i tabellen hvor attributtet gender ikke er null

- `select count(distinct firstname) from person;`

  gir antall forskjellige verdier i attributtet firstname (**null** telles ikke med)

min() og max()

- min(attributt) og max(attributt) gir henholdsvis minste og største verdi av attributtet

min() og max()

- min(attributt) og max(attributt) gir henholdsvis minste og største verdi av attributtet
- Attributtet må være numerisk eller tekstlig
  (date og time håndteres som tekststrenger)

min() og max()

- min(attributt) og max(attributt) gir henholdsvis minste og største verdi av attributtet
- Attributtet må være numerisk eller tekstlig
  (date og time håndteres som tekststrenger)
- Eksempel: Gitt tabellen Ansatt(anr, navn, lonn, avd). Finn den største lønnsforskjellen i salgsavdelingen:

```
select max(lonn) - min(lonn)
from Ansatt
where avd = 'Salg';
```

min() og max()

- min(attributt) og max(attributt) gir henholdsvis minste og største verdi av attributtet
- Attributtet må være numerisk eller tekstlig
  (date og time håndteres som tekststrenger)
- Eksempel: Gitt tabellen Ansatt(anr, navn, lonn, avd). Finn den største lønnsforskjellen i salgsavdelingen:

```
select max(lonn) - min(lonn)
from Ansatt
where avd = 'Salg';
```

- Merk at det *ikke* er lov å ha regneuttrykk som parameter i min() og max()

sum() og avg()

- sum(attributt) og avg(attributt) beregner henholdsvis summen og gjennomsnittet av verdiene i attributtet

sum() og avg()

- sum(attributt) og avg(attributt) beregner henholdsvis summen og gjennomsnittet av verdiene i attributtet
- Attributtet må være numerisk

sum() og avg()

- sum(attributt) og avg(attributt) beregner henholdsvis summen og gjennomsnittet av verdiene i attributtet
- Attributtet må være numerisk
- Tupler hvor attributtet er **null**, blir ignorert. (Dette er viktig for avg())
  - For *alle* aggregeringer unntatt `count(*)`!

sum() og avg()

- sum(attributt) og avg(attributt) beregner henholdsvis summen og gjennomsnittet av verdiene i attributtet
- Attributtet må være numerisk
- Tupler hvor attributtet er **null**, blir ignorert. (Dette er viktig for avg())
  - For *alle* aggregeringer unntatt `count(*)`!
- Eksempel: Gitt tabellen Ansatt(anr, navn, lonn, avd). Finn sum lønnsutgifter og gjennomsnittslønn for salgsavdelingen:

```
select sum(lonn), avg(lonn)
from Ansatt
where avd = 'salg';
```

Gruppering: Eksempel

Finn antall ansatte i hver avdeling og gjennomsnittlig lønn for disse:

Ansatt(anr, navn, lønn, avd)
Avdeling(avdnr, a-navn, leder)

```
select    a-navn, count(*), avg(lønn)
from      Ansatt, Avdeling
where     avd = avdnr -- joinbetingelse
group by  a-navn;
```

group by (gruppering)

- Gruppering er å dele forekomstene inn i grupper og så gi **en** resultatlinje for hver gruppe

group by (gruppering)

- Gruppering er å dele forekomstene inn i grupper og så gi **en** resultatlinje for hver gruppe
- Syntaksen er slik:

```
select <resultatattributt-liste>
from <tabell-liste>
where <betingelse>
group by <grupperingsattributt-liste>;
```

- Resultatet beregnes slik:

group by (gruppering)

- Gruppering er å dele forekomstene inn i grupper og så gi **en** resultatlinje for hver gruppe
- Syntaksen er slik:

```
select <resultatattributt-liste>
from <tabell-liste>
where <betingelse>
group by <grupperingsattributt-liste>;
```

- Resultatet beregnes slik:
  1. Beregn **select** * **from** <tabell-liste> **where** <betingelse>

group by (gruppering)

- Gruppering er å dele forekomstene inn i grupper og så gi **en** resultatlinje for hver gruppe
- Syntaksen er slik:

```
select <resultatattributt-liste>
from <tabell-liste>
where <betingelse>
group by <grupperingsattributt-liste>;
```

- Resultatet beregnes slik:
  1. Beregn **select** * **from** <tabell-liste> **where** <betingelse>
  2. Lag grupper av de tuplene som er like i alle grupperingsattributtene

group by (gruppering)

- Gruppering er å dele forekomstene inn i grupper og så gi **en** resultatlinje for hver gruppe
- Syntaksen er slik:

```
select <resultatattributt-liste>
from <tabell-liste>
where <betingelse>
group by <grupperingsattributt-liste>;
```

- Resultatet beregnes slik:
  1. Beregn **select** * **from** <tabell-liste> **where** <betingelse>
  2. Lag grupper av de tuplene som er like i alle grupperingsattributtene
  3. Utfør aggregeringsfunksjonene lokalt innenfor hver gruppe og presenter én resultatlinje for hver gruppe

group by (gruppering)

- Gruppering er å dele forekomstene inn i grupper og så gi **en** resultatlinje for hver gruppe
- Syntaksen er slik:

```
select <resultatattributt-liste>
from <tabell-liste>
where <betingelse>
group by <grupperingsattributt-liste>;
```

- Resultatet beregnes slik:
  1. Beregn **select** * **from** <tabell-liste> **where** <betingelse>
  2. Lag grupper av de tuplene som er like i alle grupperingsattributtene
  3. Utfør aggregeringsfunksjonene lokalt innenfor hver gruppe og presenter én resultatlinje for hver gruppe
- En god regel er å inkludere alle grupperingsattributtene i resultatattributt-listen.

group by (gruppering)

- Gruppering er å dele forekomstene inn i grupper og så gi **en** resultatlinje for hver gruppe
- Syntaksen er slik:

```
select <resultatattributt-liste>
from <tabell-liste>
where <betingelse>
group by <grupperingsattributt-liste>;
```

- Resultatet beregnes slik:
  1. Beregn **select** * **from** <tabell-liste> **where** <betingelse>
  2. Lag grupper av de tuplene som er like i alle grupperingsattributtene
  3. Utfør aggregeringsfunksjonene lokalt innenfor hver gruppe og presenter én resultatlinje for hver gruppe
- En god regel er å inkludere alle grupperingsattributtene i resultatattributt-listen.
- Litt avhengig av databasen hva som skjer ellers...

Og nå... en liten Powerpoint

WHERE vs. HAVING

- **where**-betingelsen velger ut de tuplene som skal danne datagrunnlaget for grupperingen

WHERE vs. HAVING

- **where**-betingelsen velger ut de tuplene som skal danne datagrunnlaget for grupperingen
- **having**-betingelsen plukker ut de tuplene fra det ferdig-grupperte resultatet som skal med i det endelige svaret

WHERE vs. HAVING

- **where**-betingelsen velger ut de tuplene som skal danne datagrunnlaget for grupperingen
- **having**-betingelsen plukker ut de tuplene fra det ferdig-grupperte resultatet som skal med i det endelige svaret
- **having**-betingelsen kan inneholde aggregatfunksjoner, men det kan ikke **where**-betingelsen

WHERE vs. HAVING

- **where**-betingelsen velger ut de tuplene som skal danne datagrunnlaget for grupperingen
- **having**-betingelsen plukker ut de tuplene fra det ferdig-grupperte resultatet som skal med i det endelige svaret
- **having**-betingelsen kan inneholde aggregatfunksjoner, men det kan ikke **where**-betingelsen
- Siden **having** håndterer en (mye) mindre datamengde enn **where**, er det i kompliserte spørringer lurt å legge så mye som mulig av logikken inn i **having**-betingelsen

Generelt utseende av SQL-spørsmål

```
  select    [distinct] <resultatattributter>
  from      <tabeller>
[ where     <utvalgbetingelse> ]
[ group by <grupperingsattributter>
[ having    <resultatbetingelse> ] ]
[ order by <ordningsattributter> ]
```

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
3. Grupper resultattuplene i henhold til like verdier i grupperingsattributtene angitt i **group by**-klausulen

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
3. Grupper resultattuplene i henhold til like verdier i grupperingsattributtene angitt i **group by**-klausulen
4. Beregn aggregeringer (count, sum,. . .) per gruppe

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
3. Grupper resultattuplene i henhold til like verdier i grupperingsattributtene angitt i **group by**-klausulen
4. Beregn aggregeringer (count, sum,. . .) per gruppe
5. Fjern de gruppene som ikke oppfyller resultatbetingelsen i **having**-klausulen

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
3. Grupper resultattuplene i henhold til like verdier i grupperingsattributtene angitt i **group by**-klausulen
4. Beregn aggregeringer (count, sum,. . .) per gruppe
5. Fjern de gruppene som ikke oppfyller resultatbetingelsen i **having**-klausulen
6. Behold bare attributtene i **select** (projeksjon)

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
3. Grupper resultattuplene i henhold til like verdier i grupperingsattributtene angitt i **group by**-klausulen
4. Beregn aggregeringer (count, sum,. . .) per gruppe
5. Fjern de gruppene som ikke oppfyller resultatbetingelsen i **having**-klausulen
6. Behold bare attributtene i **select** (projeksjon)
7. Fjern flerforekomster hvis **select distinct**

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
3. Grupper resultattuplene i henhold til like verdier i grupperingsattributtene angitt i **group by**-klausulen
4. Beregn aggregeringer (count, sum,. . .) per gruppe
5. Fjern de gruppene som ikke oppfyller resultatbetingelsen i **having**-klausulen
6. Behold bare attributtene i **select** (projeksjon)
7. Fjern flerforekomster hvis **select distinct**
8. Sorter i henhold til **order by**