



UNIVERSITETET  
I OSLO

# SQL

## Structured Query Language

(forts.)

# null

- Resultatet av å evaluere et uttrykk som produserer en ***skalar verdi***, kan være **null**
- Mulige tolkninger av **null**:
  - Verdien fins, men er ukjent
  - Verdi her har ingen mening
  - Verdi fins, men er privilegert informasjon

# Regler for null

- **null** som del av et aritmetisk uttrykk, gir **null** som svar
- Sammenlikning av **null** med en verdi, gir **unknown** som svar
  - unntak: **is distinct from** (som er omtrent som  $\langle \rangle$ , men sammenlikner som om **null** var en verdi)
- Det er ikke lov å bruke **null** eksplisitt som del av et uttrykk
- Vi kan spørre om resultatet av en beregning er **null**:  
**A is null**  
**A is not null**

# Gruppering og aggregering med null

- **null** ignoreres i aggregeringer – bortsett fra i `count(*)`  
Eksempel:  
Hvis A er attributt i relasjonen R, vil
  - **select count(A) from R**  
gi antall rader i R hvor A ikke er **null**
  - **select count(\*) from R**  
gi antall rader i R  
(inklusive de som er **null** i alle attributter)
- **null** behandles som en *ordinær* verdi ved gruppering

# unknown

- **unknown and true = unknown**  
**unknown and false = false**  
**unknown or true = true**  
**unknown or false = unknown**  
**not unknown = unknown**
- Vi får ikke bruke **unknown** eksplisitt som del av et uttrykk
- Hvis en betingelse evalueres til **unknown** for et tuppel, vil tuppelet ikke komme med i svaret
- Vi kan spørre om resultatet av en beregning er **unknown**:  
**A is unknown**  
**A is not unknown**

# Relasjonssammenligninger – I

- SQL har fem operatører som sammenligner med innholdet i en hel relasjon:
  - **exists** R      (betyr  $\exists$  forekomst i R)
  - **in** R            (betyr  $\in$  R)
  - **not in** R        (betyr  $\notin$  R)
  - **any** R            (betyr en vilkårlig verdi i R)
  - **all** R             (betyr alle verdier i R)

# Relasjonssammenligninger – II

- **any** og **all** brukes i praksis bare på relasjoner med ett attributt
  - **some** er synonym for **any**
  - **= any** er ekvivalent med **in**
- Eksempel:  
V.pris < **all** (        **select** R.verdi  
                                 **from** .....  
                                 **where** ..... )

Dette betyr at V.pris skal være mindre enn den minste R.verdi vi fant i delspørsmålet (sub-queriet)

# Relasjonssammenligninger – III

- **[not] in** kan brukes på ett attributt eller på en liste av attributter

- Eksempler:

V.knr **not in (select kunde  
                                  from .... where .... )**

(A.navn, 'frisør') **in (select navn, yrke  
                                  from .... where .... )**



# Relasjonssammenligninger – IV

- **exists (select \* from R)**  
betyr at  $\exists t (t \in R)$ ,  
dvs. at ekstensjonen til R ikke er tom
- Med andre ord: **exists** er SQLs eksistenskvantor
- SQL har ingen tilsvarende allkvantor
- Vi bruker formelen
$$\forall t(P(t)) \Leftrightarrow \neg(\neg\forall t(P(t))) \Leftrightarrow \neg(\exists t(\neg P(t)))$$
- Dette betyr at vi kan uttrykke at betingelsen C holder for alle tupler i R slik:  
**not exists (select \* from R where not C)**

# Nestede select-setninger

- **select**-setninger kan nestes til vilkårlig dybde
- Eksempel:  
Finn alle filmtitler som har vært brukt i to eller flere filmer.

```
select distinct title  
from Movie Old  
where year < any  
  (select year  
   from Movie  
   where title = Old.title);
```

# Skopregler

- Et attributt i en subquery tilhører en av tuppelvariablene i subqueryet hvis en av disse har dette attributtet
- Hvis ikke, søkes attributtet i nærmeste omsluttende (sub)querys tuppelvariable, osv
- Skopregelen kan brytes ved å kvalifisere attributtet med navnet på en tuppelvariabel fra en omsluttende query

# Kostbare operasjoner i SQL

- **distinct:**  
Sortering er generelt kostbart  
Bør brukes med forsiktighet
- **union, intersect, except:**  
SQL beregner set-variantene av disse  
(dvs. at flerforekomster fjernes)  
Vurder å bruke **union all, intersect all, except all** som er bag-variantene

# SQLs DML

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

# insert

**insert into** R(A1, A2, ..., Ak)  
**values** (v1, v2, ..., vk);

**insert into** R(A1, A2, ..., Ak)  
**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**  $A_1=E_1, \dots, A_k=E_k$

**[where C];**

**delete from** R

**[where C];**

# Dat typer i SQL

Datatype	Forklaring
integer	Heltall
smallint	Heltall, mindre plasskrevende enn integer
real, float	Flyttall (synonyme betegnelser)
double precision	Flyttall med høyere presisjon
numeric(p,s)	p signifikante sifre, s sifre etter desimalpunktum
char(n), varchar(n)	Tekst med henholdsvis fast og variabel lengde
text	Tekst med vilkårlig lengde. Ikke-standard SQL!
bit(n), bit varying(n)	Bitstreng med henholdsvis fast og variabel lengde
boolean	Boolsk verdi
date	Dato
time	Klokkeslett
timestamp	Dato og klokkeslett



# SQLs DDL

- **create**: Opprette tabell
- **drop**: Fjerne tabell
- **alter table**: Endre tabell

Herunder:

- Legge til eller fjerne kolonner
- Legge til eller fjerne indekser
- Legge til, fjerne eller endre integritetsregler (constraints)

# create

**create table R**

(A<sub>1</sub> type\_1 [skranke\_1],

...

A<sub>n</sub> type\_n [skranke\_n],  
[LISTE AV SKRANKER]

);

# 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

# Indekser

DBMS-avhengig syntaks:

**create index X on R(A<sub>1</sub>,...,A<sub>k</sub>);**

**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 og tidkrevende

# Vurdering av indeksbruk

Anta at StarsIn(movieTitle, movieYear, starName) har størrelse 10 blokker  
Anta at det i middel er 3 stjerner i hver film,  
og at hver stjerne spiller i 3 filmer

Q<sub>1</sub>: **select** movieTitle, movieYear **from** StarsIn **where** starName=s;

Q<sub>2</sub>: **select** starName **from** StarsIn **where** movieTitle=t **and** movieYear=y;

I: **insert into** StarsIn **values** (t,y,s);

p<sub>i</sub>: Andel av tiden benyttet på Q<sub>i</sub>, i=1,2, så andel tid brukt på I er 1-(p<sub>1</sub>+p<sub>2</sub>)  
Tallene i tabellen angir antall disk-aksesser

Indeks:	Ingen	starName	movieTitle	Begge
Q <sub>1</sub>	10	4	10	4
Q <sub>2</sub>	10	10	4	4
I	2	4	4	6
Snitt:	2+8p <sub>1</sub> +8p <sub>2</sub>	4+6p <sub>2</sub>	4+6p <sub>1</sub>	6-2p <sub>1</sub> -2p <sub>2</sub>

# SQLs VDL

**create view** VIEWNAVN **as**  
SELECT-SETNING;

**drop view** VIEWNAVN;

# Modifisering av viewdata

- **Oppdaterbare views**  
Views hvor det er mulig å oversette modifikasjon på tupler i viewet til modifikasjoner på tupler i basisrelasjonene
- Forenklet: Et view er oppdaterbart hvis det er definert ved en **select** (*ikke distinct*) på attributter fra en basisrelasjon eller et annet oppdaterbart view R hvor
  - **where** ikke involverer R i et subquery
  - listen i **select** er fyldig nok til at vi kan etterfylle ikke angitte attributter med defaultverdier eller **null** og slik konstruere et basistuppel som produserer det angitte viewtuppelet

# SQLs DCL

**grant** PRIVILEGIER  
**on** DATABASEELEMENT  
**to** BRUKERLISTE  
**[with grant option]**

**revoke** PRIVILEGIER  
**on** DATABASEELEMENT  
**from** BRUKERLISTE

- Vi tar detaljene i neste forelesning



# Øvingsoppgave 3

- Skjema:

Prosjekt(P#, Pnavn, Kunde, Pleder, StartDato)

Ansatt(A#, Navn, Tittel, Fdato, Pnr, AnsDato)

Timeliste(A#, Dato, P#, Timer)

Kunde(K#, Knavn, Adresse)

- Oppgave:

Finn navn, totalt antall timer utført på, og startdato for alle prosjekter bestilt av kunden «Pust og pes AS». Sorter dem slik at det nyeste prosjektet kommer først.

# Øvingsoppgave 3

- Prosjekt(P#, Pnavn, Kunde, Pleder, StartDato)  
Ansatt(A#, Navn, Tittel, Fdato, Pnr, AnsDato)  
Timeliste(A#, Dato, P#, Timer)  
Kunde(K#, Knavn, Adresse)
- Løsning:  
**select** Pnavn, SUM(Timer) **as** TimeSum, StartDato  
**from** Kunde K, Timeliste T, Prosjekt P  
**where** Knavn = 'Pust og pes AS' **and**  
K# = P.Kunde **and** P.P# = T.P#  
**group by** P.P#, Pnavn, StartDato  
**order by** StartDato **desc**