UNIVERSITETET
I OSLO

SQL

Structured Query Language

SQL

- **SQL – Structured Query Language –**
  er et *deklarativt* språk for spørringer mot relasjonsdatabaser
- Uttrykkskraften er omtrent som den i relasjonsalgebraen utvidet med tilleggsoperatorene
- 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/8.4/interactive/

SQL-standarder

- Flere standarder:
  - ANSI SQL (1986)
  - SQL2 (SQL-92) = oppdatert versjon av ANSI SQL
  - SQL:1999 (SQL3) = SQL2 + objekt-relasjonelle egenskaper mm
  - SQL:2003 = utvidet versjon av SQL:1999
- Mange dialekter:
  - **Hvert DBMS har sine særegenheter**
  - Alle oppfyller ANSI SQL, det meste av SQL2 og litt av SQL:1999 og SQL:2003
  - 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

3-skjemaarkitekturen for databaser (repetisjon)

- **Presentasjonslaget**
  beskrevet med *eksterne skjemaer* («*views*»)– hvordan informasjon skal presenteres for ulike brukere
- **Det logiske laget**
  beskrevet i *det begrepsmessige skjemaet* – hva som kan lagres og hva som er lovlige forandringer
- **Det fysiske laget**
  beskrevet i *det interne skjemaet* – hvordan informasjon lagres, forandres og bearbeides

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 …
  betyr at leddet kan gjentas vilkårlig mange ganger
- S **union** T
- S **intersect** T
- S **except** T
- ...

Tuppelvariabeltolkning av select-setningen

1. Tilordne en variabel til hver relasjon i **from**
   Variablene kalles ***tuppelvariable***
2. La hver tuppelvariabel systematisk gjennomløpe tuplene i den tilhørende relasjonen
   (f.eks. ved nestede løkker, en for hver tuppelvariabel)
   Plukk ut de tuplene der **where-**betingelsen holder
3. Grupper de utvalgte tuplene i henhold til grupperingsattributtene i **group by**
4. Velg ut de gruppene som oppfyller betingelsen i **having**
5. Sorter som beskrevet av **order by**
6. Aggreger og velg ut attributter for fremvisning i henhold til beskrivelsen i **select**
7. Hvis **select distinct**, fjern flerforekomster (duplikater) fra resultattuplene

Relasjonsalgebratolkning av select-setningen

- Hvis vi ser nøye på vanlige **select**-setninger, er det ofte to vesensforskjellige typer setningsledd i **where**-klausulen:
  - Seleksjon av tupler i en relasjon
  - Join av tupler på tvers av relasjoner

Relasjonsalgebratolkning av select-setningen (forts.)

For å få et uttrykk som er rimelig effektivt beregnbart, vil databasesystemet oversette til et uttrykk som likner det vi får ved å gjøre følgende:

1. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
2. Selekter ifølge seleksjonsbetingelsene i **where**
3. Grupper i henhold til beskrivelsen i **group by**
   Aggreger i henhold til aggregeringsbeskrivelsene i **select** og **having**
4. Velg ut grupper ifølge betingelsen i **having**
5. Sorter i henhold til **order by**
6. Lag en liste av uttrykk for (utvidet) projeksjon ifølge **select**
7. Fjern flerforekomster hvis **select distinct**

Klassiske operatorer fra relasjons-algebraen uttrykt i SQL – I

Algebra	**SQL**
$R \cup S$	(**select** * **from** R) **union** (**select** * **from** S);
$R \cap S$	(**select** * **from** R) **intersect** (**select** * **from** S);
$R - S$	(**select** * **from** R) **except** (**select** * **from** S);
$R \times S$	(**select** * **from** R) **cross join** S;
	select * **from** R, S;

Husk krav om unionkompatibilitet for R og S i **union**, **intersect**, **except**!

Klassiske operatorer fra relasjons-algebraen uttrykt i SQL – II

Algebra	**SQL**
$\sigma_C(R)$	**select** * **from** R **where** C;
$\pi_L(R)$	**select distinct** L **from** R;
$R \overset{}{\underset{C}{\bowtie}} S$	**select** * **from** R **join** S **on** C;
$R \bowtie S$	**select** * **from** R **natural join** S;

Klassiske operatorer fra relasjons-algebraen uttrykt i SQL - III

Algebra	**SQL**
$\rho_S(R)$	**select** * **from** R **as** S;
$\rho_{S(B_1,...,B_n)}(R)$	det nærmeste vi kommer, er **select** A_1 **as** B_1, ..., A_n **as** B_n **from** R **as** S;

Bagoperatorer fra relasjonsalgebraen uttrykt i SQL

Algebra	**SQL**
$R \cup S$	(**select** * **from** R) **union all** (**select** * **from** S);
$R \cap S$	(**select** * **from** R) **intersect all** (**select** * **from** S);
$R - S$	(**select** * **from** R) **except all** (**select** * **from** S);
$\pi_L(R)$	**select** L **from** R;
resten	som for klassisk SQL

Tilleggsoperatorer fra relasjons-algebraen uttrykt i SQL

Algebra	**SQL**
$\delta(R)$	**select distinct * from** R;
$\gamma_L(R)$	**select** L **from** R **group by** <grupperingsattributtene i L>;
$\tau_L(R)$	**select** * **from** R **order by** L;
$R \overset{o}{\bowtie} S$	**select** * **from** R **natural full outer join** S;
$R \overset{o}{\bowtie}_L S$	**select** * **from** R **natural left outer join** S;
$R \overset{o}{\underset{C}{\bowtie}} S$	**select** * **from** R **full outer join** S **on** C;

Gruppering

- ***Gruppering*** er å uttrykke relasjonsoperatoren $\gamma_L(R)$ i SQL
- Grupperingsattributtene fra L plasseres i **group by**-klausulen
- Både grupperingsattributter og aggregerings-uttrykk fra L plasseres i **select**-klausulen

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 priviligert 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 <>, 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 operatorer 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

- **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 m
where year < any (select year
                  from Movie
                  where title = m.title);
```

Skopregler

- Et (ikke-kvalifisert) 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

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)

Indekser

DBMS-avhengig syntaks:
create index X **on** R(A_1,..,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 og tidkrevende

Vurdering av indeksbruk

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

Q_1: **select** movieTitle, movieYear **from** StarsIn **where** starName=s;
Q_2: **select** starName **from** StarsIn **where** movieTitle=t **and** movieYear=y;
I: **insert into** StarsIn **values** (t,y,s);

p_i: Andel av tiden benyttet på Q_i, i=1,2, så andel tid brukt på I er $1–(p_1+p_2)$
Tallene i tabellen angir antall diskaksesser

Indeks:	Ingen	starName	movieTitle	Begge
Q_1	10	4	10	4
Q_2	10	10	4	4
I	2	4	4	6
Snitt:	$2+8p_1+8p_2$	$4+6p_2$	$4+6p_1$	$6-2p_1-2p_2$

SQLs VDL

create view viewnavn **as**
select .. **from** ..;

drop view viewnavn;

SQLs DCL

- Hver bruker har en **autorisasjonsID**
  Hver ID tilordnes **privilegier**:
  - **select**: Tillat select på noen attributter i en relasjon
  - **insert**: Tillat insert på noen attributter i en relasjon
  - **update**: Tillat update på noen attributter i en relasjon
  - **delete**: Tillat delete på en relasjon
  - **references**: Tillat bruk av en relasjon i en integritetsregel (assertion/attributtbasert/tuppelbasert)
  - **usage**: Tillat bruk av et databaseelement
  - **trigger**: Tillat definisjon av triggere mot en relasjon
  - **execute**: Tillat eksekvering av PSM-kodebiter
  - **under**: Tillat opprettelse av subtyper

Privilegier

- Privilegier tildeles ved
  grant <privilegier> **on** <databaseelement>
  to <brukere> [**with grant option**];
  og fratas ved
  revoke <privilegier> **on** <databaseelement>
  from <brukere> [**cascade** | **restrict**]

Oppgaver …

…. finner dere i læreboken og på

http://www.sqlzoo.net/

(der er det løsningsforslag også …)

Ekstramateriale/ Repetisjon fra INF1300

Select-setningens enkeltdeler – I

- **select**
  Angir hvilke attributter som skal vises i svaret (også aggregeringsinformasjon)
- **distinct**
  Fjerner flerforekomster (duplikater) av svar-tuplene
- **from**
  Navn på de relasjonene spørringen refererer til
- **where**
  Seleksjonsbetingelse
  (kan inneholde en eller flere join-betingelser)

Select-setningens enkeltdeler – II

- **group by**
  Angir grupperingsattributter til bruk ved aggregering
- **having**
  Angir en betingelse på resultatet av grupperingen; velger ut noen av gruppene
  Kan aggregere som del av utvelgelsesprosessen
- **order by**
  Ordner tuplene i henhold til angitte kriterier (kriterier som kan inngå i ordningen, er alle attributter, aggregeringer, algebraiske uttrykk over attributter,...)

Uttrykk i betingelser – I

- **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**}
  - **null**-test: P **is null** eller P **is not null**
  - Relasjonssammenlikning: **exists**, **in**, **all**, **any**

Uttrykk i betingelser – II

- 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 – III

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

Aggregering

- Aggregeringer i **select** tar formen ***agg***(A), eventuelt ***agg***(A) **as** B for å navngi resultatattributtet
  hvor ***agg*** er en av aggregeringsoperatorene **sum**, **avg**, **max**, **min**, **count**
  Eks: **sum**(A), **count**(A)
  Andre former: **count**(*), **agg**(**distinct** A)
- I en **select** med gruppering kan *bare* attributter nevnt i **group by** forekomme uaggregert i **select**-klausulen

Gruppering med having-klausul

- Klausulen i **having** er en betingelse (et boolsk uttrykk) satt sammen av attributter fra relasjonene i **from**
- Aggregering i **having** anvendes på gruppene dannet i henhold til tilhørende **group by**-klausul
- Bare attributtene i **group by** kan forekomme uaggregert i **having**
- Virkemåte: Etter gruppering beregnes betingelsen for hver enkelt gruppe som et hele Bare de gruppene som oppfyller betingelsen, tas med i den videre beregningen

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 renavnes til B i resultatrelasjonen

Relasjonssammenligninger – I

- **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** ….)