UNIVERSITETET I OSLO

SQL

Structured Query Language

(The intergalactic dataspeak)

SQL

- **SQL – Structured Query Language –** er et *deklarativt* språk for spørringer mot relasjonsdatabaser
- Uttrykkskraften er omtrent som den i relasjonsalgebraen utvidet med tilleggs-operatorene
- SQL inneholder også konstruksjoner for å definere nye relasjonsdatabaser, for å legge inn og endre data i databasen, mm

Hvordan uttale «SQL»

- På norsk uttaler vi SQL bokstav for bokstav: «ess–ku–ell»
- På engelsk uttales SQL «'si:kwel»
  Årsaken er historisk:
  - SQL ble utviklet av IBM, og prototypen het SEQUEL – et akronym for «Structured English QUEry Language»
  - Da SEQUEL ble lansert som et produkt i 1976, ble navnet forkortet til SQL, men uttalen ble beholdt og har overlevd til nå

SQL-standarder

- Flere standarder:
  - ANSI SQL (1986)
  - SQL2 (SQL-92)
  - SQL3 (SQL:1999) = SQL2 + objekt-relasjonelle egenskaper mm
- Mange dialekter:
  - Hvert DBMS har sine særegenheter
  - Alle oppfyller ANSI SQL, stort sett også SQL2
  - 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 konseptuelle (eller 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

Select-setningen

- Typisk utseende:
  select $a_1, a_2, ..., a_j$
  from $r_1, r_2, ..., r_k$
  where C;
  hvor $r_1, r_2, ..., r_k$ er relasjonsnavn
  $a_1,...,a_j$ er attributter fra $r_1, r_2, ..., r_k$
  C er en betingelse
- Essensielt uttrykker konstruksjonen det samme som relasjonsalgebrauttrykket
  $\pi_{a1,a2,...,aj}(\sigma_C(r_1 \times r_2 \times ... \times r_k))$

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

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 grupperings-attributtene i **group by**
   Velg ut de gruppene som oppfyller betingelsen i **having**
4. Aggreger og velg ut attributter for fremvisning i henhold til beskrivelsen i **select**
   Hvis **select distinct**, fjern flerforekomster (duplikater) fra resultattuplene
5. Sorter som beskrevet av **order by**

Relasjonsalgebratolkning av select-setningen

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

Merknader til SQL

- SQL skiller ikke mellom store og små bokstaver, unntatt i tekststrenger
- SQL beregner bager
  (med unntak av noen av operatorene)

Klassiske operatorer fra relasjons-algebraen uttrykt i SQL – I

Algebra	SQL
$R \cup S$	R **union** S
$R \cap S$	R **intersect** S
$R - S$	R **except** S
$R \times S$	R **cross join** S

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
$R \overset{C}{\bowtie} S$	R **join** S **on** C
$R \bowtie S$	R **natural join** S

Klassiske operatorer fra relasjons-algebraen uttrykt i SQL - III

Algebra	SQL
$\rho_S(R)$	**select** * **from** R **as** S
$\rho_{S(B1,...,Bn)}(R)$	**select** A1 **as** B1, ..., An **as** Bn **from** R **as** S

Bagoperatorer fra relasjonsalgebraen uttrykt i SQL

Algebra	SQL
$R \cup S$	R **union all** S
$R \cap S$	R **intersect all** S
$R - S$	R **except all** S
$\pi_L(R)$	**select** L **from** R
andre	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
$R \overset{o}{\bowtie} S$	R **natural full outer join** S
$R \overset{o}{\bowtie}_L S$	R **natural left outer join** S
$R \overset{o}{\underset{C}{\bowtie}} S$	R **full outer join** S **on** C

Gruppering – I

- ***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

Gruppering – II

- Aggregeringer i **select** tar formen **agg**(A), eventuelt **agg**(A) **as** B for å navngi resultatattributtet
  (**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 aggregeringer 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

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** ∈ {=, <, >, <=, >=, <>, **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 neste forelesning)

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'

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

Øvingsoppgave 1

- 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 og tittel på alle som har arbeidet på prosjektet 'Vintersalg'

Øvingsoppgave 2

- 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 på alle som har arbeidet på et prosjekt ledet av Else Brun,
  og sorter dem etter stigende ansiennitet

Flere oppgaver …

…. finner dere i læreboken og på

http://www.sqlzoo.net/

(der er det løsningsforslag også …)