SELECT-kommandoen

Hva gjør kommandoen?

Henter ut informasjon fra en gitt database, f.eks, jeg vil hente fødselsdato fra persondatabase og vil da få oversikt alle personers fødselsdatoer.

Formen til SELECT:

SELECT <kolonner>
FROM <tabeller>

- hvor <kolonner> er en liste med kolonne-navn
- og <tabeller> er en liste med tabell-navn

Resultatet av en SELECT-spørring er alltid en ny tabell, som består av:

- kolonnene i <kolonner>
- basert på radene i tabellene i <tabeller>

Slik ser en spørring med SELECT-kommandoen ut under:

Velge en enkelt kolonne

Spørring som henter ut alle navn i `Customer`-tabellen

```
SELECT Name
  FROM Customer
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

Velge flere kolonner

Spørring som henter alle navn -og fødselsdato-par i `Customer`-tabellen

```
SELECT Name, Birthdate
  FROM Customer
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

- Her så er ‘Name-kolonnen’ markert som skyldes av at i spørringen, så var vi ute etter alle navnene fra denne “Customer” tabellen.
- Går ann å hente ut flere kolonner, så til høyre, er både “Name-” og “Birthdate-kolonnen” markert som igjen skyldes av spørringen.

Velge alle kolonner

Spørring som henter alle kolonnene i `Customer`-tabellen

```
SELECT *
  FROM Customer
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

- For å hente alle kolonner i en tabell, skriv SELECT *, der stjerne henter alle kolonner

Uttrykk i SELECT

- Vi kan anvende uttrykk i SELECT-kommandoen, hvor hensikten er å manipulere verdiene, f.eks:

```
SELECT product_name, unit_price * 8
FROM products
```

  Hva denne spørringen gjør, er at den henter ut alle produktnavn hvor prisen blir ganget med 8, så unit_pricen blir da endret for hver produkt. Men ved å gjøre dette så vil resultat-tabellen ha et navn som er ukjent hvor det vil stå ?column? Her må vi da bruke AS, for å gi den resultat-tabellen et navn. Definisjonen for AS, er nevnt lenger nede.

Duplikater

- Duplikater av og til uønsket så vi kan da fjerne dem gjennom DISTINCT-nøkkelordet. Dette blir seende slikt:

```
SELECT DISTINCT contact_title
FROM customers
WHERE contact_title LIKE '%Manager%'
```

  I dette tilfelle så vil få unike Managers og ikke duplikater.

Aggregering

- En aggregeringsfunksjon er en funksjon som returnerer en enkel verdi fra en samling verdier, i SQL har vi mange slike funksjoner men de vi skal benytte er da **sum,avg og count**. Disse kan bli anvendt på alle verdier i en kolonne eller på en gruppe av rader.
- Går ann å kombinere ulike aggregeringsfunksjoner, hvor de returnerer en enkel verdi. Kan da ikke kombinere denne med andre kolonner i samme SELECT-klausul da disse gir ut en verdi, ikke tabeller og informasjon knyttet til den tabellen.
- **sum:** For å summere en hel kolonne, kan vi putte sum(<column>) i SELECT-klausulen. Formen blir da slikt:

```
SELECT sum(units_in_stock) AS total_nr_products
FROM products
```

- **avg:** Gjennomsnitt
- **max:** Maksimum
- **min:** Maksimum
- **count:** Antall rader, vil da oppgi antall rader som tilfredsstiller en gitt WHERE, betingelse.

WHERE-kommandoen:

Hva gjør kommandoen?

Kommandoen velger de radene vi ønsker å hente, de spesifikke radene vi er interesserte i. Kan sees som en if-test fra ulike programmeringsspråk hvor du manipulerer dataen som ble "SELECT" til å passe dine formål. F.eks, er jeg interessert i å finne fødselsdatoen til alle personer med alder 20. Denne sjekken vil da utføres i WHERE-kommandoen. Siden WHERE-kommandoen er som if-test fra ulike programmeringsspråk, så vil AND, OR være gyldige å bruke også for denne kommandoen.

Formen til WHERE:

SELECT <kolonner>
FROM <tabeller>
WHERE <betingelse>

- <Betingelse> er et uttrykk over kolonnenavnene fra tabellene.
- For hver rad evalueres dette uttrykket til sant eller usant.
- Resultatet er det samme som før, men begrenset til kun de radene som gjør <betingelse> sann.

Velge ut spesifikke rader

Spørring som gir fødselsdatoen til kunden ved navn John Mill

```
SELECT Birthdate
  FROM Customer
 WHERE Name = 'John Mill'
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

- Her ser vi at 'John Mill' sin rad blir markert, som skyldes av at vi var ute etter hans navn gjennom spørringen. Merk at siden det står "SELECT Birthdate", så betyr det at vi henter "John Mill" sin bursdagsdato, som er 1986-11-16, ikke hele raden som omfatter andre attributter som ikke ble nevnt i "SELECT"-spørringen.

Bruke både AND og OR

Spørring som finner navn på kunder som har kjøpt mindre enn 5 eller mer enn 15 produkter og er født etter '2000-01-01'

```
SELECT Name FROM Customer
 WHERE (NrProducts < 5 OR
        NrProducts > 15) AND
       Birthdate > '2000-01-01'
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

- Som if-betingelser i Java og python, kan dette også benyttes i SQL hvor vi kan en eller flere betingelser med AND, OR.

Velge TVer med `LIKE`

Spørring som finner navn, pris og merke på alle TVer

```
SELECT Name, Brand, Price
  FROM Product
 WHERE Name LIKE 'TV%'
```

Resultat

ProductID	Name	Brand	Price	Stock
0	TV 50 inch	Sony	8999	29
1	Laptop 2.5GHz	Lenovo	7499	12
2	Laptop 8GB RAM	HP	6999	80
3	Speaker 500	Bose	4999	42
4	TV 48 inch	Panasonic	11999	31
5	Phone S6	IPhone	5195	65

- Her ser vi at vi kan anvende 'LIKE' kommandoen og '%' i betingelsen vår, hvor i dette tilfellet så markeres alle rader som har TV i starten av strengen

Regulære uttrykk

- LIKE støtter kun % (og _ for wildcard enkelt karakter)
- Ønsker man komplisert matchin kan man bruke SIMILAR TO
- SIMILAR TO bruker litt rar miks av LIKE-syntaks (%) og vanlige regulære uttrykk
- F.eks. er Name = 'abc' et mulig svar for

  SELECT Name

  FROM Products

  WHERE Name SIMILAR TO '%(b|d)%' (Samme tankegang som ved 1150)

Negasjon

- Av og til vil vi bare ha svar som ikke tilfredsstiller et uttrykk
- Bruker da NOT-nøkkelordet, samme som (" != ") ved python og java
- F.eks;

  SELECT Name

  FROM Products

  WHERE NOT Description LIKE '%simple%'

  er sant for alle rader som ikke har ordet 'simple' i sin Description
- Merk at

  NOT (E1 AND E2) er ekvivalent med (NOT E1) OR (NOT E2)

  NOT (E1 OR E2) er ekvivalent med (NOT E1) AND (NOT E2)

NULL

- Samme tankegang som ved python og java, når vi vil lagre variabler, men ikke har lagt til en verdi for den gitte variabelen/objektet, så kan vi angi dem til å være NULL, hvor vi kan etterhvert legge til data senere.

Students

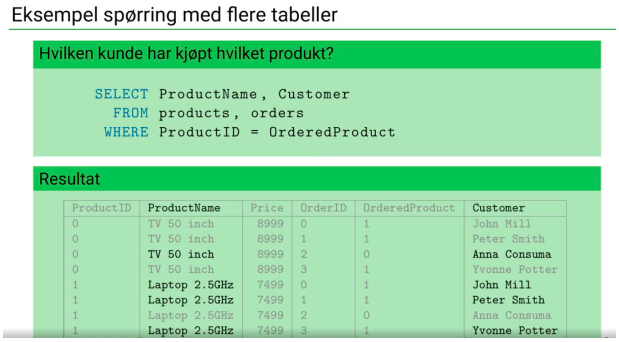
SID	StdName	StdBirthdate
0	Anna Consuma	1978-10-09
1	Anna Consuma	1978-10-09
2	Peter Young	2009-03-01
3	Carla Smith	1986-06-14
4	Sam Penny	NULL

FROM-kommandoen:

Hva gjør kommandoen?

Henter en tabell fra en gitt database, spesifiserer hvilken tabell man er interessert i å hente ut. F.eks, jeg er interessert i å hente Person-tabellen eller Vare-tabellen, osv.

Kombinere flere tabeller i en spørring (JOIN)



Det går ann å kombinere flere tabeller i en spørring. Som vist over så er de radene markert for de som oppfyller WHERE-betingelsen som består av to verdier fra forskjellige tabeller. Slike spørringer kalles “JOINS”, hvor vi har spørringer over flere tabeller. Det finnes ulike type “JOINS” som vi kan lage:

- **Cross-join(Kryssproduktet)** mellom t1 og t2 ser slikt ut:

  SELECT * FROM t1, t2
- **Equi-join(Ekvivalent)** mellom t1 og t2 ser slikt ut:

  SELECT * FROM t1, t2

  WHERE t1.a = t2.b
- **Theta-join** mellom t1 og t2

  SELECT * FROM t1, t2

  WHERE <theta>(t1.a, t2.b)

  hvor <theta> er en/annen relasjon (f,eks. <, = , !=, LIKE) eller mer komplisert uttrykk.

Like kolonnenavn

- Ved bruk av join, så kan det hende at flere tabeller i samme spørring, kan ha kolonner med like verdier, f.eks at Student og Ansatt har begge Navn som verdi. Da må vi spesifisere hvem sitt navn det er ved å bruke tabellnavnet som prefiks. Altså, Student.navn og Ansatt.navn, ved WHERE-betingelsen. Akkurat som ved java og python hvor vi må skrive objektnavnet så spesifisere til dets klassevariabler.

Relasjonell algebra og SQL

- SQL-spørringene med joins kan også oversettes til relasjonsalgebra

- Eksempel er vist under:

  SELECT <columns>

  FROM <t1>, <t2>, ….., <tN>

  WHERE <condition>

Notasjon for joins

- Vi har INNER JOIN og ON, som er nøkkelord som beskriver de spørringene over.
- Fremfor å skrive:

  SELECT product_name

  FROM products AS p, orders AS o

  WHERE p.product_id = o.product_id AND o.unit_price > 700

  kan man skrive:

  SELECT p.product_name

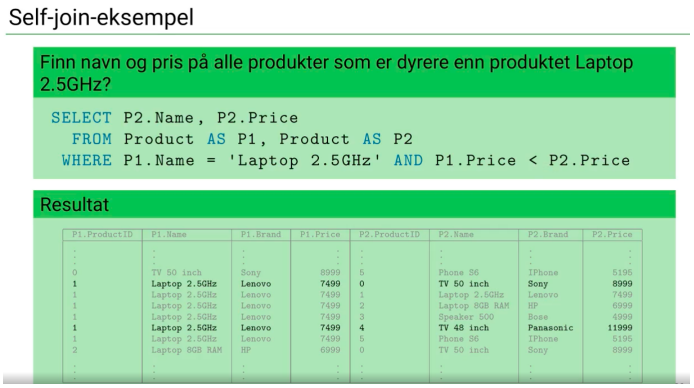
  FROM products AS p INNER JOIN order_details AS o

  ON (p.product_id = o.product_id)

  WHERE o.unit_price > 7000

Self-joins

- For å kombinere informasjon fra rader i samme tabell, kan man benytte self-join, hvor man bruker den samme tabellen to eller flere ganger i FROM-klausulen. Man må gi dem forskjellige navn for at dette skal funke.



Forskjellen mellom SQL og Relasjonsalgebra

- Men i den relasjonsmodellen er relasjonene mengder av tupler
- I en mengde kan et element kun forekomme én gang, f.eks.:

Person

Navn	Alder
Per	13
Ola	24
Mari	13
Karl	25
Ida	25

$\pi_{Alder}(Person)$

Alder
13
24
25

- I SQL har vi tabeller i stedet for relasjoner (multi-mengder av tupler):

```
SELECT Alder
  FROM Person
```

⇒

Alder
13
24
13
25
25

SQL kan lagre duplikater -> multimengder

CREATE-kommandoen: Lager en ny tabell, eller noe avhengig av hva du vil lage, f,eks lage en tabell om mat.

INSERT-kommandoen: Setter inn rader i en tabell

UPDATE-kommandoen: Oppdaterer data i en tabell

DELETE-kommandoen: Sletter rader fra en tabell

DROP-kommandoen: Sletter en hel ting, f.eks en hel tabell

JOIN-kommandoen: Gjør det mulig å kombinere ulike rader fra diverse tabeller sammen.

AS: Benyttes for å navngi ulike tabeller eller verdier. F.eks, FROM “order_details” AS “o”, betyr at “order_details” nå heter “o” slik at vi slipper å benytte “order_details” ut spørringen men kan bruke “o” istedenfor.

\e: Gjør at vi kan formatere spørringen vår ved å anvende en editor. For å gå ut av editoren så gjør vi følgende: Trykk på ESC, deretter tast “: wq”

\d: Gjør at vi for oversikt over en database, for eksempel, \d product, for oversikt over skjemaet til produkt, som omfatter alle nøkler og attributter for den tabellen og relasjonene i tabellen.

LIKE og “%” / wildcard

Hva gjør dette?

Gjør at vi kan hente du de attributtene med hva som er angitt mellom prosenten, altså, jeg har lyst til å hente ut alle aldre som er 16. Ved å skrive ‘% 16 %‘, så vil alle personer med alder, 16 fås ut

Formen til LIKE og “%” / wildcard

Name LIKE ‘TV%’

- Sant for alle Name-verdier som starter med ‘TV’.
- f.eks, ‘TV 50 inch’ og ‘TVSHOW’, ser at vi skriver ut alle strenger med ‘TV’ i starten.
- Men ikke f.eks, ‘hello’ eller ‘MTV’, siden de ikke har ‘TV’ helt på starten av strengen.

Name LIKE ‘%TV’

- Sant for alle Name-verdier som slutter med ‘TV’.

- f.eks, ‘50 inch TV’ og ‘MTV’, ser at vi skriver ut alle strenger med ‘TV’ mot slutten.
- Men ikke f.eks, ‘TV2’ eller ‘Fun TV program’, siden de ikke har ‘TV’ mot slutten av strengen

Name LIKE ‘%TV%’’

- Sant for alle Name-verdier som inneholder ‘TV’ i strengen, plassering kan være hvor som helst.
- f.eks, ‘50 inch TV’ og ‘TV2’, ser at plasseringen ikke har en rolle.
- Men ikke f.eks, ‘T2V’ eller ‘hello’, siden de ikke har ‘TV’ i strengen sin.

Name LIKE ‘%TV%inch’’

- Sant for alle Name-verdier som inneholder ‘TV’ i strengen og slutter med ‘inch’.
- f.eks, ‘TV 50 inch’ og ‘Fun TV program pinch’, ser at plasseringen ikke har en rolle.
- Men ikke f.eks, ‘TV 50 inches’ eller ‘50 inch TV’.