IN2090 – Databaser og datamodellering

09 – Aggregering i grupper

Leif Harald Karlsen
leifhka@ifi.uio.no

Universitetet i Oslo

Aggregere i grupper

- Vi har sett hvordan vi kan aggregere over hele kolonner
- Det finner derimot en egen klausul for å gruppere radene før man aggregerer
- Nemlig `GROUP BY <kolonner>`
- `GROUP BY` tar en liste med kolonner, og grupperer dem i henhold til likhet på verdiene i disse kolonnene
- Vi kan så bruke aggregeringsfunksjoner på hver gruppe i `SELECT`-klausulen
- Vi kan da også ha de grupperende kolonnene sammen med aggregatet i `SELECT`-klausulen
- Kun de grupperte kolonnene gir mening å ha utenfor et aggregat i `SELECT`

Aggregere i grupper: Eksempel

Finn gjennomsnittsprisen for hver kategori

```
  SELECT Category, avg(Price) AS Averageprice
    FROM Products
GROUP BY Category
```

Resultat

Products

ProductID (int)	Name (text)	Brand (text)	Price (float)	Category (text)
0	TV 50 inch	Sony	8999	Televisions
1	Laptop 2.5GHz	Lenovo	7499	Computers
2	Laptop 8GB RAM	HP	6999	Computers
3	Speaker 500	Bose	4999	Speakers
4	TV 48 inch	Panasonic	11999	Televions
5	Laptop 1.5GHz	IPhone	5195	Computers

Aggregere i grupper: Eksempel

Finn gjennomsnittsprisen for hver kategori

```
SELECT Category, avg(Price) AS Averageprice
  FROM Products
GROUP BY Category
```

Resultat: Velg ut kolonner og grupper ihht. `Categories`

Price	Category
8999	Televisions
7499	Computers
6999	Computers
4999	Speakers
11999	Televisions
5195	Computers

Aggregere i grupper: Eksempel

Finn gjennomsnittsprisen for hver kategori

```
SELECT Category, avg(Price) AS Averageprice
  FROM Products
GROUP BY Category
```

Resultat: Regn ut aggregatet for hver gruppe og ferdigstill

avg(Price)	Category
10499	Televisions
6731	Computers
4999	Speakers

Aggregering i grupper: Eksempel 1

Finn antall produkter per bestilling

```
  SELECT order_id, sum(quantity) AS nr_products
    FROM order_details
GROUP BY order_id;
```

Aggregering i grupper: Eksempel 2

Finn gjennomsnittspris for hver kategori (i Northwind)

```
  SELECT c.category_name, avg(p.unit_price) AS Averageprice
    FROM categories AS c INNER JOIN products AS p
         ON (c.category_id = p.category_id)
GROUP BY c.category_name;
```

Gruppere på flere kolonner

- Vi kan også gruppere på flere kolonner
- Da vil hver gruppe bestå av de radene med like verdier på alle kolonnene vi grupperer på

Finn antall produkter for hver kombinasjon av kategori og hvorvidt produktet fortsatt selges

```
  SELECT c.category_name, p.discontinued, count(*) AS nr_products
    FROM categories AS c INNER JOIN products AS p
         ON (c.category_id = p.category_id)
GROUP BY c.category_name, p.discontinued;
```

Aggregering i grupper: Eksempel 3

Finn navn på ansatte og antall bestillinger den ansatte har håndtert, sortert etter antall bestillinger fra høyest til lavest

```
  SELECT format('%s %s', e.first_name, e.last_name) AS emp_name,
         count(*) AS num_orders
    FROM orders AS o INNER JOIN employees AS e
         ON (o.employee_id = e.employee_id)
GROUP BY e.first_name, e.last_name
ORDER BY num_orders DESC;
```

Filtrere på aggregat-resultat

- I enkelte tilfeller er vi kun interessert i grupper hvor et aggregat har en bestemt verdi
- F.eks. dersom man vil vite kategorinavn og antall produkter på de kategoriene som har flere enn 10 produkter
- Nå kan vi gjøre dette med en delspørring:

```
SELECT category_name, nr_products
FROM (
  SELECT c.category_name, count(*) AS nr_products
  FROM categories AS c
       INNER JOIN products AS p ON (c.category_id = p.category_id)
  GROUP BY c.category_name) AS t
WHERE nr_products > 10;
```

- Men det finnes en egen klausul for å velge ut grupper

HAVING-klausulen

- Denne klausulen heter `HAVING` og kommer rett etter `GROUP BY`, slik:

```
  SELECT c.category_name, count(*) AS nr_products
    FROM categories AS c
         INNER JOIN products AS p ON (c.category_id = p.category_id)
GROUP BY c.category_name
  HAVING count(*) > 10;
```

- Merk: Kan ikke bruke navnene vi gir i `SELECT`
- `HAVING` blir altså evaluert på hver gruppe
- Fungerer altså som en slags `WHERE` for grupper

Oversikt over SQLs `SELECT`

- Vi har nå sett mange nye klausuler
- Generelt ser våre SQL-spørringer nå slik ut:

```
    WITH <navngitte-spørringer>
  SELECT <kolonner>
    FROM <tabeller>
   WHERE <uttrykk>
GROUP BY <kolonner>
  HAVING <uttrykk>
ORDER BY <kolonner> [DESC]
   LIMIT <N>
  OFFSET <M>
```

- I denne rekkefølgen (`LIMIT` og `OFFSET` kan bytte plass)
- Kan selvfølgelig droppe klausuler, men må ha `GROUP BY` for å ha `HAVING`

Hvor kan ulike navn brukes

- Navnene vi lager med `AS` i `WITH`-klausulen kan brukes i alle de etterfølgende spørringene
- Navnene fra `SELECT` kan brukes i `ORDER BY`-klausulen og alle ytre spørringer
- Navnene fra `FROM` kan brukes i alle klausuler utenom samme `FROM`-klausul

Takk for nå!

Neste video gjennomgår avanserte eksempler.