SQL-yhteenvetofunktiot

- SQL:ssä joukko yhteenvetofunktioita (aggregate function, koostefunktio)
  - AVG keskiarvo
  - MIN pienin arvo (minimi)
  - MAX suurin arvo (maksimi)
  - SUM summa
  - COUNT lukumäärä
  - Yhteenvetofunktioita käytettäessä tulosriviä ei muodostetakaan jokaisesta valintaehdon täyttävästä riviyhdistelmästä vaan ellei ryhmittelyä ole määritelty muodostetaan yksi tulosrivi koko aineistosta

SQL-yhteenvetofunktiot

- Opiskelijoiden lukumäärä:
  - select count(*) from opiskelija;
- Count:n parametrina voisi käyttää myös mitä tahansa vakiota, tulos olisi sama eli rivien lukumäärä
  - select count(1) from opiskelija;
- Jos parametrina annetaan sarake saadaan siinä olevien ei-tyhjien arvojen määrä
  - select count(onumero) from opiskelija;

SQL-yhteenvetofunktiot

- Milloin opiskeluajaltaan vanhin Helsinkiläinen opiskelija on aloittanut opintonsa?

```
select min(aloitusvuosi) from opiskelija
where kaupunki='Helsinki';
```

- Keskiarvoa, summaa, minimiä ja maksimia laskettaessa tyhjäarvot jätetään huomioimatta
- Keskimääräinen opintoviikkomäärä
  - select avg(opintoviikot) from kurssi;

SQL-yhteenvetofunktiot

- Keskiarvoa ja summaa laskettaessa on yleensä syytä ottaa mukaan kaikki sarakkeen arvot
- Lukumääriä laskettaessa voi erilaisten arvojen laskeminenkin tulla kyseeseen
- Monellako eri paikkakunnalla opiskelijat asuvat?
  - select count(distinct kaupunki) from opiskelija;

SQL-yhteenvetofunktiot

- Yhteenvetofuntion sisältävään kyselyyn ei saa mukaan dataa niiltä yksittäisiltä riveiltä, joilta funktio lasketaan
- Minkä kurssien opintoviikkomäärä on suurin?
- Ei siis ole mahdollista:

```
select nimi, max(opintoviikot)
from kurssi;
```

yksityiskohta

SQL-yhteenvetofunktiot

```
Select nimi, opintoviikot from kurssi
where opintoviikot>= ALL
(select opintoviikot from kurssi);
```

- antaa oikean vastauksen
- Seuraavan pitäisi myös toimia:

```
select nimi,maksi
from kurssi,
     (select max(opintoviikot) maksi from kurssi) as m
where opintoviikot =maksi;
```

alikysely

SQL-yhteenvetofunktiot

- Jos kyselyyn liitetään ryhmittelymääre (Group by) muuttuu tuolosrivien muodostusperiaate jälleen:
  - **muodostetaankin yksi tulosrivi kutakin ryhmää kohti**
  - group by määreessä luetellaan sarakkeet, joiden perusteella ryhmittely tehdään
  - kaikki ne rivit, joilla on sama arvo luetelluissa sarakkeissa muodostavat ryhmän
  - ryhmät muodostetaan sen jälkeen kun on ensin sovellettu where-ehtoa rivien karsintaan.

SQL-yhteenvetofunktiot

- Group by A
- Taulu X

A	B	C	D	
1	4	6	7	Ryhmä a=1
1	1	4	2	
1	5	5	2	
2	4	8	7	Ryhmä a=2
2	3	5	1	
3	1	5	2	
3	2	4	6	

SQL-yhteenvetofunktiot

- Select A, sum(B) from X group by A;

A	B
1	10
2	7
3	3

Group by -lausetta käytettäessä tulostietoluettelossa voi olla yhteenvetofunktioiden lisäksi vain niitä sarakkeita, jotka esiintyvät group by -lauseessa. (kaikkien ei tarvitse olla mukana, mutta yleensä ne ovat)

SQL-yhteenvetofunktiot

- Kurssien harjoitusryhmiin ilmoittautuneiden opiskelijoiden lukumäärät

```
select nimi, ryhmanro, count(*)
from kurssi, ilmoittautuminen
where ilmoittautuminen.kurssikoodi=
        kurssi.koodi
group by nimi, ryhmanro;
```

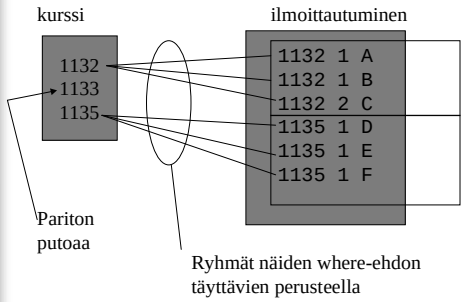
Ongelma: tyhjät ryhmät eivät tule mukaan!
Miksi?

SQL-yhteenvetofunktiot

```
select nimi, ryhmanro, count(*)
from kurssi, ilmoittautuminen
where ilmoittautuminen.kurssikoodi=kurssi.koodi
group by nimi, ryhmanro
union
(select nimi, ryhmanro, 0
from kurssi, harjoitusryhma H,
where koodi=H.kurssikoodi and
      (koodi,H.ryhmanro) not in
(select kurssikoodi, ryhmanro
from ilmoittautuminen));
```

- tuottaa tyhjätkin - Miksi näin monimutkaista?

SQL-yhteenvetofunktiot



SQL-yhteenvetofunktiot

- Ulkoliitosta käyttäen kysely onnistuisi myös

```
select nimi, h.ryhmanro, count(distinct opisknro)
from kurssi,
     harjoitusryhma h left outer join ilmoittautuminen i
     on h.kurssikoodi= i.kurssikoodi and
          h.ryhmanro=I.ryhmanro
where kurssi.koodi= h.kurssikoodi
group by nimi, h.ryhmanro
```

  - tässä ei voi käyttää count(*) koska tyhjästäkin ryhmästä tulee rivi

SQL-yhteenvetofunktiot

- Ryhmäkohtaisen rivin mukaanottamista tulokseen voidaan rajoittaa having määreellä.
- Having-ehto kuten where-ehto, mutta se perustuu ryhmäkohtaisesti lasketun yhteenvetofunktion arvoon

SQL-yhteenvetofunktiot

- Ryhmät, joihin on ilmoittautunut yli 20 opiskelijaa

```
select nimi, ryhmanro, count(*)
from kurssi, ilmoittautuminen
where ilmoittautuminen.kurssikoodi=kurssi.koodi
group by nimi, ryhmanro
having count(*) >20;
```

SQL-yhteenvetofunktiot

- Ryhmät muodostetaan aina where-ehdon soveltamisen jälkeen
- Käytännössä ryhmien muodostaminen tarkoittaa tulostaulun järjestämistä ryhmitysattribuuttien perusteella
- Order by -järjestäminen suoritetaan viimeiseksi

SQL-yhteenvetofunktiot

  - Harjoitusryhmät ilmoittautumismäärän mukaan laskevassa järjestyksessä

```
select nimi, ryhmanro, count(*)
from kurssi, ilmoittautuminen
where ilmoittautuminen.kurssikoodi=kurssi.koodi
group by nimi, ryhmanro
order by count(*) desc;
```

SQL-yhteenvetofunktiot

- Millä kurseilla on suurin keskimääräinen ryhmäkoko

```
select nimi, H.ryhmanro, max(avg(count(*)))
from kurssi, harjoitusryhma H, ilmoittautuminen I
where koodi=H.kurssikoodi and
     H.kurssikoodi=I.kurssikoodi and
     H.ryhmanro=I.ryhmanro
group by nimi, H.ryhmanro
     *
Ajettuna Oraclessa:   VIRHE rivillä 1:
ORA-00937: tämä ei ole yhden ryhmän koostefunktio
```

YHTEENVETOFUNKTIOITA EI VOI KETJUTTAA

SQL-yhteenvetofunktiot

Kurssin ryhmien koot (vain ei-tyhjät ryhmät)

```
select kurssikoodi,ryhmanro,count(*) lkm
from ilmoittautuminen
group by kurssikoodi, ryhmanro
```

tuotaa testikannassa:

KURSSIKOODI	RYHMANRO	LKM
1134	1	6
1134	2	4
1134	3	3
1135	1	6
1135	2	5
1135	3	2
1136	1	6
1137	1	5

SQL-yhteenvetofunktiot

- Kurssien keskimääräiset ryhmäkoot

alikysely

```
select koodi, nimi, avg(lkm)
 from kurssi,
   (select kurssikoodi,ryhmanro,count(*) lkm
    from ilmoittautuminen
    group by kurssikoodi, ryhmanro)  A
     where koodi= A.kurssikoodi
 group by koodi, nimi
```

SQL-yhteenvetofunktiot

Testikannasta ed. kysely tuo tuloksen

KOODI	NIMI	AVG(LKM)
1134	Informaatiojärjestelmät	4,33333333
1135	Java ohjelmointi	4,33333333
1136	Oliotietokannat	6
1137	Tietorakenteet	5

SQL-yhteenvetofunktiot

Edellinen kysely

```
select koodi, nimi, avg(lkm)
from kurssi,
    (select kurssikoodi,ryhmanro,count(*) lkm
     from ilmoittautuminen
     group by kurssikoodi, ryhmanro)  A
    where koodi= A.kurssikoodi
group by koodi, nimi
having avg(lkm)>= ALL
  (select avg(lkm)
   from kurssi,
     (select kurssikoodi,ryhmanro,count(*) lkm
      from ilmoittautuminen
      group by kurssikoodi, ryhmanro)  A
     where koodi= A.kurssikoodi
     group by koodi)
```

SQL-yhteenvetofunktiot

- Tulos samasta testikannasta

KOODI	NIMI	AVG(LKM)
1136	Oliotietokannat	6

SQL-yhteenvetofunktiot

```
select   avg(lkm)
      from kurssi,
        (select kurssikoodi,ryhmanro,count(*) lkm
         from ilmoittautuminen
         group by kurssikoodi, ryhmanro)  A
        where koodi= A.kurssikoodi
        group by koodi
```

- tulos tästä on

```
AVG(LKM)
----------
4,33333333
4,33333333
6
5
```

Ryhmitysperuste koodi ei ole mukana tuloksessa

SQL-yhteenvetofunktiot

- Huom. edelliset kyselyt toimivat täysin oikein vain, jos kaikkiin ryhmiin on ilmoittautuneita, sillä tyhjät ryhmät putoavat pois jo ilmoittautujalukumäärien laskennasta.
- Kovin monimutkaisia kyselyjä, kuten edellinen, voi helpottaa käyttämällä näkymiä (views) eli johdettuja tauluja

SQL-Näkymät (views)

- **Näkymä** on kyselyn avulla määritelty taulu.
- Määrittely:
  create view *taulunimi* [(*sarakkeet*)] as *kysely*;
- sarakkeet on luettelo sarakenimiä
  - jos luettelo on mukana, siinä täytyy olla yhtä monta nimeä kuin kyselyssä tulossarakkeita
  - jos luettelo puuttuu, sarakkeet nimetään kyselyn mukaisesti

SQL-Näkymät (views)

- Esim.

```
Create view tyhja_ryhma (kurssikoodi, nimi, ryhmanro) as
  select kurssikoodi, nimi, ryhmanro
  from kurssi, harjoitusryhma
  where koodi= kurssikoodi and
    (kurssikoodi, ryhmanro) not in
    (select kurssikoodi, ryhmanro
     from ilmoittautuminen)
```

- Data näkymään saadaan muista tauluista.

SQL-Näkymät (views)

- Näkymän tietoja ei tallenneta kantaan.
- Kyselyn kohdistuessa näkymään järjestelmä muokkaaa kyselyä siten, että se kohdistuukin näkymän määrittelevässä kyselyssä oleviin tauluihin. Esim.

```
Select * from tyhja_ryhma
```

- aiheuttaisi sen määrittelevän kyselyn suorituksen

SQL-Näkymät (views)

- Miksi näkymiä:
  - suojaussyyt
    - käyttäjälle voidaan antaa käyttöoikeus näkymään perustaulun asemesta
    - näin voidaan rajoittaa oikeus esim. vain joihinkin riveihin
    - Oletetaan että opetunnus on opettajan käyttäjätunnus
    - Funktio **user** antakoon kulloisenkin käyttäjän tunnuksen.
    - Määritellään näkymä:

```
create view oma_kurssi as
select * from kurssi where luennoija=user;
```

    - voidaan määritellä 'grant select, update on oma_kurssi to public' vaikka kuka tahansa voi tehdä kyselyjä näkymään perustuen on taulu tyhjä muille kuin luennoijille

SQL-Näkymät (views)

  - Kyselyjen yksinkertaistaminen:
    - piilotetaan monimutkaisuus näkymämäärittelyyn
    - Aiemmin esillä ollut kysely ryhmien ilmoittautujamäärät voitaisiin tyhja-ryhma -näkymää käyttäen esittää:

```
select nimi, ryhmanro, count(*)
from kurssi, ilmoittautuminen
where ilmoittautuminen.kurssikoodi=kurssi.koodi
group by nimi, ryhmanro
union
(select nimi, ryhmanro, 0
 from tyhja_ryhma)
```

SQL-Näkymät (views)

- Edelläolevan voisi määritellä näkymäksi, jolloin kyselyjen teko on jatkossa todella helppoa

```
create view ryhmäkoko (nimi, ryhma, lkm) as
   select nimi, ryhmanro, count(*)
   from kurssi, ilmoittautuminen
   where ilmoittautuminen.kurssikoodi=kurssi.koodi
   group by nimi, ryhmanro
   union    (select nimi, ryhmanro, 0
              from tyhja_ryhma);

select * from ryhmakoko;
```

SQL-Näkymät (views)

- Tietoriippumattomuus
  - Kun ohjelman tietokantaoperaatiot perustuvat näkymiin voidaan tietokannan rakennetta muuttaa, eikä ohjelmaan tarvitse koskea lainkaan - muutetaan vain näkymämäärittelyä siten, että aiemman kaltainen näkymä saadaan myös muuttuneesta kannasta

SQL-Näkymät (views)

- Näkymiä voi käyttää kyselyissä kuten tauluja.
- Myös tietokannan ylläpito näkymien kautta on rajoitetusti mahdollista.

SQL - Tietokannan ylläpito

- SQL sisältää operaatiot tietokannan sisällön muodostamiseen ja ylläpitoon:
- insert - uusien rivien vienti tauluun
- delete - rivien poisto
- update - rivien muutos

SQL - Tietokannan ylläpito

- Insert lauseella on kaksi muotoa:
- insert into *taulu* [(*sarakenimet*)] values (*arvot*)
  - tällä muodolla lisätään yksi rivi ja arvot annetaan vakioina tai vakioihin perustuvina lausekkeina
- insert into *taulu* [(*sarakenimet*)] *kysely*
  - *tällä muodolla kyselyn tulosrivit lisätään tauluun*

SQL - Tietokannan ylläpito

```
CREATE TABLE kurssi (
        koodi  numeric(8) NOT NULL ,
        nimi  varchar(40) NOT NULL ,
        opintoviikot  numeric(5,1) NOT NULL ,
        luennoija  varchar(12) NOT NULL,
        PRIMARY KEY (koodi ),
        FOREIGN KEY (luennoija) REFERENCES
  opettaja)

insert into kurssi values
  (1234,'Tietokantojen perusteet',2,'HLAINE');
```

  - lisää tauluun kokonaisen rivin

SQL - Tietokannan ylläpito

- Jos luennoijaa ei tiedetä voidaan lisäys tehdä seuraavasti:

```
insert into kurssi values
  (1234,'Tietokantojen perusteet',2,NULL); tai

insert into kurssi (koodi, nimi, opintoviikot)
  values (1234,'Tietokantojen perusteet',2);
```

- **sarakeluetteloa käytetään siis silloin kun annetaan vain osa sarakkeista**

SQL - Tietokannan ylläpito

- Jos luennoijan tunnusta ei tiedetä, voitaisiin lisäys tehdä seuraavasti:

```
insert into kurssi
  select (1234, 'Tietokantojen perusteet', 2,opetunnus)
  from opettaja
  where nimi='Laine Harri' ;
```

– Tämä toimii odotetusti, jos kannassa on vain yksi tämän niminen opettaja, muuten lisäys kaatuu avaimen yksikäsitteisyysvirheeseen, sillä lisäys epäonnistuu, jos se rikkoo eheysehtoja

SQL - Tietokannan ylläpito

- Kurssille 'Ohjelmoinnin perusteet' ilmoittautuneiden opiskelijoiden siirto kurssin 'Java-ohjelmointi' vastaaviin ryhmiin:

```
CREATE TABLE ilmoittautuminen (
      kurssikoodi numeric(8) not null,
      ryhmanro numeric(2) not null,
      opisknro  numeric(5) NOT NULL ,
      ilm_aika  date NOT NULL ,
      PRIMARY KEY (opisknro, kurssikoodi) ,
      FOREIGN KEY (kurssikoodi, ryhmanro)  REFERENCES
          harjoitusryhma on delete cascade,
      FOREIGN KEY (opisknro) REFERENCES opiskelija )
```

SQL - Tietokannan ylläpito

Oracle-SQL:llä

```
Insert into ilmoittautuminen
  select java.kurssikoodi, ryhmanro, opisknro, sysdate
  from kurssi java, kurssi ohpe, ilmoittautuminen
  where java.nimi='Java-ohjelmointi' and
        ohpe.nimi='Ohjelmoinnin perusteet' and
        ohpe.koodi=ilmoittautuminen.kurssikoodi;
```

SQL - Tietokannan ylläpito

- Rivien muutokset (update)

update *taulu*
set *sarake1=lauseke1* [, …]
[where *kohteen rajausehdot*]

– samalla kertaa voi muttaa useita sarakkeita,
– muutetaan kaikki where-ehdon täyttävär rivit
– jos ehto puuttuu muutetaan kaikki taulun rivit

SQL - Tietokannan ylläpito

- Muutetaan kurssin Java-ohjelmointi opintoviikkomäärä kolmeksi

```
update kurssi
set opintoviikot=3
where nimi='Java ohjelmointi';
```

– Muutos epäonnistuu, jos se rikkoo eheysehtoja.

SQL - Tietokannan ylläpito

- Rivien poisto (delete)

```
delete from taulu
[where poistettavien rajausehdot ];
```

  - Poistetaan kaikki ehdon täyttävät rivit
  - Jos ehto puuttuu poistetaan kaikki rivit
  - Poisto epäonnistuu jos eheysehdot rikkoutuvat (ellei muuta ole määritelty)

SQL - Tietokannan ylläpito

- Poistetaan harjoitusryhmät joihin ei ole ilmoittautuneita:

```
delete from harjoitusryhma
where (kurssikoodi, ryhmanro) not in
    (select kurssikoodi, ryhmanro
     from ilmoittautuminen);
```

SQL - Tietokannan ylläpito

- Oraclessa löytyy operaatiot taulun uudelleennimeämiseen ja kopiointiin, esim.

```
create table nimi as
select * from toinen_taulu;

rename table nimi to uusinimi;
```

- Näitä ei löydy standardista.

SQL - Tietokannan ylläpito

- Rivien siirtoa taulusta toiseen tarvitaan esimerkiksi siirrettäessä tietoja aktiivisesta taulusta historiatauluihin. Tämä suoritetaan kopioimalla (lisäämällä) rivit kohdetauluun ja sen jälkeen poistamalla ne lähtötaulusta:

```
insert into ilmohistoria
select * from ilmoittautumiset where ilm_aika<'1.1.1999';
delete from ilmoittautumiset where ilm_aika<'1.1.1999';
```

SQL - Tietokannan ylläpito

- Joissakin järjestelmissä ylläpito-operaatiot voidaan suorittaa myös näkymiin kohdistuvina. Muutos vaikuttaa silloin näkymää vastaavaan perustauluun.
- Tähän liityy rajoitteita, sillä on useita tilanteita, joissa ei pystytä yksiselitteisesti päättelemään millainen muutos perustauluihin olisi tehtävä
- Yleensä päivitettävällä näkymällä on yksi perustaulu (ei liitosta) ja taulun pääavaimen on sisällyttävä näkymään

SQL - Tietokantatapahtuma (transaktio)

- Tietokantatapahtumalla tarkoitetaan yhtenä jakamattomana kokonaisuutena pidettävää tietokantaoperaatioiden joukkoa, esimerkiksi tilisiirto:

```
update tili set saldo=saldo-500
   where tilinumero=123456;
update tili set saldo=saldo+500
   where tilinumero=654321;
```

SQL - Tietokantatapahtuma (transaktio)

- Tkhj takaa, että
  - tapahtuma suoritetaan kokonaan eikä vain osaa siitä (ei siis vain tililtäottoa)
  - ulkopuoliset näkevät vain kokonaisen tapahtuman aiheuttamat muutokset (ulkopuolinen ei voi nähdä tilannetta, jossa tililtä 123456 on otettu 500 mutta tilille 654321 ei sitä ole vielä viety)
  - tapahtuman suorituksen aikana tehdyt muutokset kantaan on peruttavissa siihen asti kunnes tapahtumaan on sitouduttu
  - kun tapahtumaan on sitouduttu (se on valmis) muutokset jäävät pysyviksi ja näkyvät myös muille.

SQL - Tietokantatapahtuma (transaktio)

- Tapahtuma päätetään onnistuneesti komennolla **commit [work]**
  - (Solidissa work on pakollinen)
- Tapahtuma voidaan päättää myös perumalla sen aikaansaamat muutokset komennolla **rollback [work]**
- Tilisiirtotapahtuma olisi siis

```
update tili set saldo=saldo-500
   where tilinumero=123456;
update tili set saldo=saldo+500
   where tilinumero=654321;
commit;
```

SQL - Tietokantatapahtuma (transaktio)

- Järjestelmät voidaan määritellä toimimaan auto-commit tilassa, jolloin jokaiseen ylläpito-operaatioon sitoudutaan välittömästi (tällöin tilisiirtoa ei voida koota transaktioksi)
- Normaalitilassa tapahtumia kuitenkin kootaan commit operaatioiden avulla. Kahden commitin välissä olevat operaatiot muodostavat tapahtuman.

SQL - Tietokantatapahtuma (transaktio)

- Yritetään poistaa tyhjät harjoitusryhmät, oletetaan, että ilmoittautumisten viiteavaimeen liittyy on delete cascade -määre

```
commit;
select count(*) from ilmoittautumiset;
>> 3500 <<
delete from harjoitusryhma
where ryhmanro is not null;
select count(*) from ilmoittautumiset;
>>> 0 <<<                          (oho!)
rollback;
select count(*) from ilmoittautumiset;
 >> 3500 <<
```

Ei ole ihan oikein

SQL-yhteenvetofunktiot

- SQL:ssä joukko yhteenvetofunktioita (aggregate function, koostefunktio)
  - AVG keskiarvo
  - MIN pienin arvo (minimi)
  - MAX suurin arvo (maksimi)
  - SUM summa
  - COUNT lukumäärä
  - Yhteenvetofunktioita käytettäessä tulosriviä ei muodostetakaan jokaisesta valintaehdon täyttävästä riviyhdistelmästä vaan ellei ryhmittelyä ole määritelty muodostetaan yksi tulosrivi koko aineistosta

SQL-yhteenvetofunktiot

- Opiskelijoiden lukumäärä:
  - select count(*) from opiskelija;
- Count:n parametrina voisi käyttää myös mitä tahansa vakiota, tulos olisi sama eli rivien lukumäärä
  - select count(1) from opiskelija;
- Jos parametrina annetaan sarake saadaan siinä olevien ei-tyhjien arvojen määrä
  - select count(onumero) from opiskelija;

SQL-yhteenvetofunktiot

- Milloin opiskeluajaltaan vanhin Helsinkiläinen opiskelija on aloittanut opintonsa?

```
select min(aloitusvuosi) from opiskelija
where kaupunki='Helsinki';
```

- Keskiarvoa, summaa, minimiä ja maksimia laskettaessa tyhjäarvot jätetään huomioimatta
- Keskimääräinen opintoviikkomäärä
  - select avg(opintoviikot) from kurssi;

SQL-yhteenvetofunktiot

- Keskiarvoa ja summaa laskettaessa on yleensä syytä ottaa mukaan kaikki sarakkeen arvot
- Lukumääriä laskettaessa voi erilaisten arvojen laskeminenkin tulla kyseeseen
- Monellako eri paikkakunnalla opiskelijat asuvat?
  - select count(distinct kaupunki) from opiskelija;

SQL-yhteenvetofunktiot

- Yhteenvetofuntion sisältävään kyselyyn ei saa mukaan dataa niiltä yksittäisiltä riveiltä, joilta funktio lasketaan
- Minkä kurssien opintoviikkomäärä on suurin?
- Ei siis ole mahdollista:

```
select nimi, max(opintoviikot)
from kurssi;
```

yksityiskohta

SQL-yhteenvetofunktiot

```
Select nimi, opintoviikot from kurssi
where opintoviikot>= ALL
(select opintoviikot from kurssi);
```

- antaa oikean vastauksen
- Seuraavan pitäisi myös toimia:

```
select nimi,maksi
from kurssi,
     (select max(opintoviikot) maksi from kurssi) as m
where opintoviikot =maksi;
```

alikysely

SQL-yhteenvetofunktiot

- Jos kyselyyn liitetään ryhmittelymääre (Group by) muuttuu tuolosrivien muodostusperiaate jälleen:
  - **muodostetaankin yksi tulosrivi kutakin ryhmää kohti**
  - group by määreessä luetellaan sarakkeet, joiden perusteella ryhmittely tehdään
  - kaikki ne rivit, joilla on sama arvo luetelluissa sarakkeissa muodostavat ryhmän
  - ryhmät muodostetaan sen jälkeen kun on ensin sovellettu where-ehtoa rivien karsintaan.

SQL-yhteenvetofunktiot

- Group by A
- Taulu X

A	B	C	D	
1	4	6	7	Ryhmä a=1
1	1	4	2	
1	5	5	2	
2	4	8	7	Ryhmä a=2
2	3	5	1	
3	1	5	2	
3	2	4	6	

SQL-yhteenvetofunktiot

- Select A, sum(B) from X group by A;

A	B
1	10
2	7
3	3

Group by -lausetta käytettäessä tulostietoluettelossa voi olla yhteenvetofunktioiden lisäksi vain niitä sarakkeita, jotka esiintyvät group by -lauseessa. (kaikkien ei tarvitse olla mukana, mutta yleensä ne ovat)

SQL-yhteenvetofunktiot

- Kurssien harjoitusryhmiin ilmoittautuneiden opiskelijoiden lukumäärät

```
select nimi, ryhmanro, count(*)
from kurssi, ilmoittautuminen
where ilmoittautuminen.kurssikoodi=
        kurssi.koodi
group by nimi, ryhmanro;
```

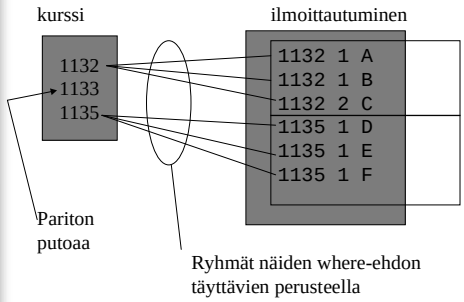
Ongelma: tyhjät ryhmät eivät tule mukaan!
Miksi?

SQL-yhteenvetofunktiot

```
select nimi, ryhmanro, count(*)
from kurssi, ilmoittautuminen
where ilmoittautuminen.kurssikoodi=kurssi.koodi
group by nimi, ryhmanro
union
(select nimi, ryhmanro, 0
from kurssi, harjoitusryhma H,
where koodi=H.kurssikoodi and
      (koodi,H.ryhmanro) not in
(select kurssikoodi, ryhmanro
from ilmoittautuminen));
```

- tuottaa tyhjätkin - Miksi näin monimutkaista?

SQL-yhteenvetofunktiot



SQL-yhteenvetofunktiot

- Ulkoliitosta käyttäen kysely onnistuisi myös

```
select nimi, h.ryhmanro, count(distinct opisknro)
from kurssi,
     harjoitusryhma h left outer join ilmoittautuminen i
     on h.kurssikoodi= i.kurssikoodi and
          h.ryhmanro=I.ryhmanro
where kurssi.koodi= h.kurssikoodi
group by nimi, h.ryhmanro
```

  - tässä ei voi käyttää count(*) koska tyhjästäkin ryhmästä tulee rivi

SQL-yhteenvetofunktiot

- Ryhmäkohtaisen rivin mukaanottamista tulokseen voidaan rajoittaa having määreellä.
- Having-ehto kuten where-ehto, mutta se perustuu ryhmäkohtaisesti lasketun yhteenvetofunktion arvoon

SQL-yhteenvetofunktiot

- Ryhmät, joihin on ilmoittautunut yli 20 opiskelijaa

```
select nimi, ryhmanro, count(*)
from kurssi, ilmoittautuminen
where ilmoittautuminen.kurssikoodi=kurssi.koodi
group by nimi, ryhmanro
having count(*) >20;
```

SQL-yhteenvetofunktiot

- Ryhmät muodostetaan aina where-ehdon soveltamisen jälkeen
- Käytännössä ryhmien muodostaminen tarkoittaa tulostaulun järjestämistä ryhmitysattribuuttien perusteella
- Order by -järjestäminen suoritetaan viimeiseksi

SQL-yhteenvetofunktiot

  - Harjoitusryhmät ilmoittautumismäärän mukaan laskevassa järjestyksessä

```
select nimi, ryhmanro, count(*)
from kurssi, ilmoittautuminen
where ilmoittautuminen.kurssikoodi=kurssi.koodi
group by nimi, ryhmanro
order by count(*) desc;
```

SQL-yhteenvetofunktiot

- Millä kurseilla on suurin keskimääräinen ryhmäkoko

```
select nimi, H.ryhmanro, max(avg(count(*)))
from kurssi, harjoitusryhma H, ilmoittautuminen I
where koodi=H.kurssikoodi and
     H.kurssikoodi=I.kurssikoodi and
     H.ryhmanro=I.ryhmanro
group by nimi, H.ryhmanro
     *
Ajettuna Oraclessa:   VIRHE rivillä 1:
ORA-00937: tämä ei ole yhden ryhmän koostefunktio
```

YHTEENVETOFUNKTIOITA EI VOI KETJUTTAA

SQL-yhteenvetofunktiot

Kurssin ryhmien koot (vain ei-tyhjät ryhmät)

```
select kurssikoodi,ryhmanro,count(*) lkm
from ilmoittautuminen
group by kurssikoodi, ryhmanro
```

tuotaa testikannassa:

KURSSIKOODI	RYHMANRO	LKM
1134	1	6
1134	2	4
1134	3	3
1135	1	6
1135	2	5
1135	3	2
1136	1	6
1137	1	5

SQL-yhteenvetofunktiot

- Kurssien keskimääräiset ryhmäkoot

alikysely

```
select koodi, nimi, avg(lkm)
 from kurssi,
   (select kurssikoodi,ryhmanro,count(*) lkm
    from ilmoittautuminen
    group by kurssikoodi, ryhmanro)  A
     where koodi= A.kurssikoodi
 group by koodi, nimi
```

SQL-yhteenvetofunktiot

Testikannasta ed. kysely tuo tuloksen

KOODI	NIMI	AVG(LKM)
1134	Informaatiojärjestelmät	4,33333333
1135	Java ohjelmointi	4,33333333
1136	Oliotietokannat	6
1137	Tietorakenteet	5

SQL-yhteenvetofunktiot

Edellinen kysely

```
select koodi, nimi, avg(lkm)
from kurssi,
    (select kurssikoodi,ryhmanro,count(*) lkm
     from ilmoittautuminen
     group by kurssikoodi, ryhmanro)  A
    where koodi= A.kurssikoodi
group by koodi, nimi
having avg(lkm)>= ALL
  (select avg(lkm)
   from kurssi,
     (select kurssikoodi,ryhmanro,count(*) lkm
      from ilmoittautuminen
      group by kurssikoodi, ryhmanro)  A
     where koodi= A.kurssikoodi
     group by koodi)
```

SQL-yhteenvetofunktiot

- Tulos samasta testikannasta

KOODI	NIMI	AVG(LKM)
1136	Oliotietokannat	6

SQL-yhteenvetofunktiot

```
select   avg(lkm)
      from kurssi,
        (select kurssikoodi,ryhmanro,count(*) lkm
         from ilmoittautuminen
         group by kurssikoodi, ryhmanro)  A
        where koodi= A.kurssikoodi
        group by koodi
```

- tulos tästä on

```
AVG(LKM)
----------
4,33333333
4,33333333
6
5
```

Ryhmitysperuste koodi ei ole mukana tuloksessa

SQL-yhteenvetofunktiot

- Huom. edelliset kyselyt toimivat täysin oikein vain, jos kaikkiin ryhmiin on ilmoittautuneita, sillä tyhjät ryhmät putoavat pois jo ilmoittautujalukumäärien laskennasta.
- Kovin monimutkaisia kyselyjä, kuten edellinen, voi helpottaa käyttämällä näkymiä (views) eli johdettuja tauluja

SQL-Näkymät (views)

- **Näkymä** on kyselyn avulla määritelty taulu.
- Määrittely:
  create view *taulunimi* [(*sarakkeet*)] as *kysely*;
- sarakkeet on luettelo sarakenimiä
  - jos luettelo on mukana, siinä täytyy olla yhtä monta nimeä kuin kyselyssä tulossarakkeita
  - jos luettelo puuttuu, sarakkeet nimetään kyselyn mukaisesti

SQL-Näkymät (views)

- Esim.

```
Create view tyhja_ryhma (kurssikoodi, nimi, ryhmanro) as
  select kurssikoodi, nimi, ryhmanro
  from kurssi, harjoitusryhma
  where koodi= kurssikoodi and
    (kurssikoodi, ryhmanro) not in
    (select kurssikoodi, ryhmanro
     from ilmoittautuminen)
```

- Data näkymään saadaan muista tauluista.

SQL-Näkymät (views)

- Näkymän tietoja ei tallenneta kantaan.
- Kyselyn kohdistuessa näkymään järjestelmä muokkaaa kyselyä siten, että se kohdistuukin näkymän määrittelevässä kyselyssä oleviin tauluihin. Esim.

```
Select * from tyhja_ryhma
```

- aiheuttaisi sen määrittelevän kyselyn suorituksen

SQL-Näkymät (views)

- Miksi näkymiä:
  - suojaussyyt
    - käyttäjälle voidaan antaa käyttöoikeus näkymään perustaulun asemesta
    - näin voidaan rajoittaa oikeus esim. vain joihinkin riveihin
    - Oletetaan että opetunnus on opettajan käyttäjätunnus
    - Funktio **user** antakoon kulloisenkin käyttäjän tunnuksen.
    - Määritellään näkymä:

```
create view oma_kurssi as
select * from kurssi where luennoija=user;
```

    - voidaan määritellä 'grant select, update on oma_kurssi to public' vaikka kuka tahansa voi tehdä kyselyjä näkymään perustuen on taulu tyhjä muille kuin luennoijille

SQL-Näkymät (views)

  - Kyselyjen yksinkertaistaminen:
    - piilotetaan monimutkaisuus näkymämäärittelyyn
    - Aiemmin esillä ollut kysely ryhmien ilmoittautujamäärät voitaisiin tyhja-ryhma -näkymää käyttäen esittää:

```
select nimi, ryhmanro, count(*)
from kurssi, ilmoittautuminen
where ilmoittautuminen.kurssikoodi=kurssi.koodi
group by nimi, ryhmanro
union
(select nimi, ryhmanro, 0
 from tyhja_ryhma)
```

SQL-Näkymät (views)

- Edelläolevan voisi määritellä näkymäksi, jolloin kyselyjen teko on jatkossa todella helppoa

```
create view ryhmäkoko (nimi, ryhma, lkm) as
   select nimi, ryhmanro, count(*)
   from kurssi, ilmoittautuminen
   where ilmoittautuminen.kurssikoodi=kurssi.koodi
   group by nimi, ryhmanro
   union    (select nimi, ryhmanro, 0
              from tyhja_ryhma);

select * from ryhmakoko;
```

SQL-Näkymät (views)

- Tietoriippumattomuus
  - Kun ohjelman tietokantaoperaatiot perustuvat näkymiin voidaan tietokannan rakennetta muuttaa, eikä ohjelmaan tarvitse koskea lainkaan - muutetaan vain näkymämäärittelyä siten, että aiemman kaltainen näkymä saadaan myös muuttuneesta kannasta

SQL-Näkymät (views)

- Näkymiä voi käyttää kyselyissä kuten tauluja.
- Myös tietokannan ylläpito näkymien kautta on rajoitetusti mahdollista.

SQL - Tietokannan ylläpito

- SQL sisältää operaatiot tietokannan sisällön muodostamiseen ja ylläpitoon:
- insert - uusien rivien vienti tauluun
- delete - rivien poisto
- update - rivien muutos

SQL - Tietokannan ylläpito

- Insert lauseella on kaksi muotoa:
- insert into *taulu* [(*sarakenimet*)] values (*arvot*)
  - tällä muodolla lisätään yksi rivi ja arvot annetaan vakioina tai vakioihin perustuvina lausekkeina
- insert into *taulu* [(*sarakenimet*)] *kysely*
  - *tällä muodolla kyselyn tulosrivit lisätään tauluun*

SQL - Tietokannan ylläpito

```
CREATE TABLE kurssi (
        koodi  numeric(8) NOT NULL ,
        nimi  varchar(40) NOT NULL ,
        opintoviikot  numeric(5,1) NOT NULL ,
        luennoija  varchar(12) NOT NULL,
        PRIMARY KEY (koodi ),
        FOREIGN KEY (luennoija) REFERENCES
  opettaja)

insert into kurssi values
  (1234,'Tietokantojen perusteet',2,'HLAINE');
```

  - lisää tauluun kokonaisen rivin

SQL - Tietokannan ylläpito

- Jos luennoijaa ei tiedetä voidaan lisäys tehdä seuraavasti:

```
insert into kurssi values
  (1234,'Tietokantojen perusteet',2,NULL); tai

insert into kurssi (koodi, nimi, opintoviikot)
  values (1234,'Tietokantojen perusteet',2);
```

- **sarakeluetteloa käytetään siis silloin kun annetaan vain osa sarakkeista**

SQL - Tietokannan ylläpito

- Jos luennoijan tunnusta ei tiedetä, voitaisiin lisäys tehdä seuraavasti:

```
insert into kurssi
  select (1234, 'Tietokantojen perusteet', 2,opetunnus)
  from opettaja
  where nimi='Laine Harri' ;
```

– Tämä toimii odotetusti, jos kannassa on vain yksi tämän niminen opettaja, muuten lisäys kaatuu avaimen yksikäsitteisyysvirheeseen, sillä lisäys epäonnistuu, jos se rikkoo eheysehtoja

SQL - Tietokannan ylläpito

- Kurssille 'Ohjelmoinnin perusteet' ilmoittautuneiden opiskelijoiden siirto kurssin 'Java-ohjelmointi' vastaaviin ryhmiin:

```
CREATE TABLE ilmoittautuminen (
      kurssikoodi numeric(8) not null,
      ryhmanro numeric(2) not null,
      opisknro  numeric(5) NOT NULL ,
      ilm_aika  date NOT NULL ,
      PRIMARY KEY (opisknro, kurssikoodi) ,
      FOREIGN KEY (kurssikoodi, ryhmanro)  REFERENCES
          harjoitusryhma on delete cascade,
      FOREIGN KEY (opisknro) REFERENCES opiskelija )
```

SQL - Tietokannan ylläpito

Oracle-SQL:llä

```
Insert into ilmoittautuminen
  select java.kurssikoodi, ryhmanro, opisknro, sysdate
  from kurssi java, kurssi ohpe, ilmoittautuminen
  where java.nimi='Java-ohjelmointi' and
        ohpe.nimi='Ohjelmoinnin perusteet' and
        ohpe.koodi=ilmoittautuminen.kurssikoodi;
```

SQL - Tietokannan ylläpito

- Rivien muutokset (update)

update *taulu*
set *sarake1=lauseke1* [, …]
[where *kohteen rajausehdot*]

– samalla kertaa voi muttaa useita sarakkeita,
– muutetaan kaikki where-ehdon täyttävär rivit
– jos ehto puuttuu muutetaan kaikki taulun rivit

SQL - Tietokannan ylläpito

- Muutetaan kurssin Java-ohjelmointi opintoviikkomäärä kolmeksi

```
update kurssi
set opintoviikot=3
where nimi='Java ohjelmointi';
```

– Muutos epäonnistuu, jos se rikkoo eheysehtoja.

SQL - Tietokannan ylläpito

- Rivien poisto (delete)

```
delete from taulu
[where poistettavien rajausehdot ];
```

  - Poistetaan kaikki ehdon täyttävät rivit
  - Jos ehto puuttuu poistetaan kaikki rivit
  - Poisto epäonnistuu jos eheysehdot rikkoutuvat (ellei muuta ole määritelty)

SQL - Tietokannan ylläpito

- Poistetaan harjoitusryhmät joihin ei ole ilmoittautuneita:

```
delete from harjoitusryhma
where (kurssikoodi, ryhmanro) not in
    (select kurssikoodi, ryhmanro
     from ilmoittautuminen);
```

SQL - Tietokannan ylläpito

- Oraclessa löytyy operaatiot taulun uudelleennimeämiseen ja kopiointiin, esim.

```
create table nimi as
select * from toinen_taulu;

rename table nimi to uusinimi;
```

- Näitä ei löydy standardista.

SQL - Tietokannan ylläpito

- Rivien siirtoa taulusta toiseen tarvitaan esimerkiksi siirrettäessä tietoja aktiivisesta taulusta historiatauluihin. Tämä suoritetaan kopioimalla (lisäämällä) rivit kohdetauluun ja sen jälkeen poistamalla ne lähtötaulusta:

```
insert into ilmohistoria
select * from ilmoittautumiset where ilm_aika<'1.1.1999';
delete from ilmoittautumiset where ilm_aika<'1.1.1999';
```

SQL - Tietokannan ylläpito

- Joissakin järjestelmissä ylläpito-operaatiot voidaan suorittaa myös näkymiin kohdistuvina. Muutos vaikuttaa silloin näkymää vastaavaan perustauluun.
- Tähän liityy rajoitteita, sillä on useita tilanteita, joissa ei pystytä yksiselitteisesti päättelemään millainen muutos perustauluihin olisi tehtävä
- Yleensä päivitettävällä näkymällä on yksi perustaulu (ei liitosta) ja taulun pääavaimen on sisällyttävä näkymään

SQL - Tietokantatapahtuma (transaktio)

- Tietokantatapahtumalla tarkoitetaan yhtenä jakamattomana kokonaisuutena pidettävää tietokantaoperaatioiden joukkoa, esimerkiksi tilisiirto:

```
update tili set saldo=saldo-500
   where tilinumero=123456;
update tili set saldo=saldo+500
   where tilinumero=654321;
```

SQL - Tietokantatapahtuma (transaktio)

- Tkhj takaa, että
  - tapahtuma suoritetaan kokonaan eikä vain osaa siitä (ei siis vain tililtäottoa)
  - ulkopuoliset näkevät vain kokonaisen tapahtuman aiheuttamat muutokset (ulkopuolinen ei voi nähdä tilannetta, jossa tililtä 123456 on otettu 500 mutta tilille 654321 ei sitä ole vielä viety)
  - tapahtuman suorituksen aikana tehdyt muutokset kantaan on peruttavissa siihen asti kunnes tapahtumaan on sitouduttu
  - kun tapahtumaan on sitouduttu (se on valmis) muutokset jäävät pysyviksi ja näkyvät myös muille.

SQL - Tietokantatapahtuma (transaktio)

- Tapahtuma päätetään onnistuneesti komennolla **commit [work]**
  - (Solidissa work on pakollinen)
- Tapahtuma voidaan päättää myös perumalla sen aikaansaamat muutokset komennolla **rollback [work]**
- Tilisiirtotapahtuma olisi siis

```
update tili set saldo=saldo-500
   where tilinumero=123456;
update tili set saldo=saldo+500
   where tilinumero=654321;
commit;
```

SQL - Tietokantatapahtuma (transaktio)

- Järjestelmät voidaan määritellä toimimaan auto-commit tilassa, jolloin jokaiseen ylläpito-operaatioon sitoudutaan välittömästi (tällöin tilisiirtoa ei voida koota transaktioksi)
- Normaalitilassa tapahtumia kuitenkin kootaan commit operaatioiden avulla. Kahden commitin välissä olevat operaatiot muodostavat tapahtuman.

SQL - Tietokantatapahtuma (transaktio)

- Yritetään poistaa tyhjät harjoitusryhmät, oletetaan, että ilmoittautumisten viiteavaimeen liittyy on delete cascade -määre

```
commit;
select count(*) from ilmoittautumiset;
>> 3500 <<
delete from harjoitusryhma
where ryhmanro is not null;
select count(*) from ilmoittautumiset;
>>> 0 <<<                          (oho!)
rollback;
select count(*) from ilmoittautumiset;
 >> 3500 <<
```

Ei ole ihan oikein