SQL kysely

Kyselyn yleisrakenne:

```
select tulostietomäärittely
    from taulukkeet
    [where valintaehdot]
    [group by ryhmitystekijät]
    [having ryhmärajoitteet]
    [order by järjestysperusta]
```

Kysely tuottaa nimettömän tulostaulun.

SQL -kyselyt

- Kyselyn from-osassa voi olla useita tauluja
- Kaikki ne taulut, joiden dataa halutaan mukaan tulokseen on annettava from-osassa
- Tauluille voidaan from-osassa antaa tilapäinen kyselyn sisäinen nimi (alias, correlation name)
  - from taulu [AS] alias
  - liitettävillä tauluilla on usein samannimisiä sarakkeita, joten taulunimeä on käytettävä tarkenteena - alias voi olla lyhenne, joka vähentää kirjoitusvaivaa

Aliasten käyttö

- Jos sama taulu esiintyy from osassa useaan kertaan, on taulun esiintymät erotettava käyttämällä aliasta
- Esim.: Kurssiparit, joilla on sama luennoija

```
select A.nimi, B.nimi
from kurssi A, kurssi B
where A.luennoija=B.luennoija and A.koodi<B.koodi
order by A.nimi, B.nimi
```

- ehto A.koodi<B.koodi estää saman parin toistumisen eri järjestyksessä

Alikyselyt

- Alikyselyllä tarkoitetaan kyselyyn upotettua toista kyselyä. Upotettua kyselyä voidaan käyttää kyselyn from osassa mutta myös where osassa valintaehtojen operandina.
- Alikyselykin tuottaa tuloksenaan taulun
- Alikyselyiden käyttöön valintaehdoissa on omia predikaatteja ja lisätarkenteita, jotka määrittelevät, miten ehdon operandia sovelletaan alikyselyn tulokseen
- IN; NOT IN; tarkenteet any,some, all; EXISTS, NOT EXISTS

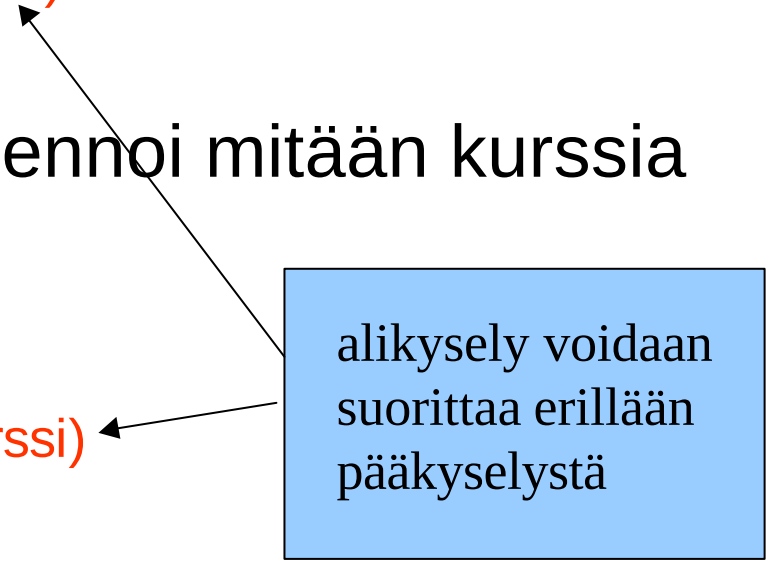
Alikyselyt

- Opettajat, jotka luennoivat jotain kurssia

```
select nimi from opettaja
where opetunnus in
   (select luennoija from kurssi)
order by nimi;
```

- Opettajat, jotka eivät luennoi mitään kurssia

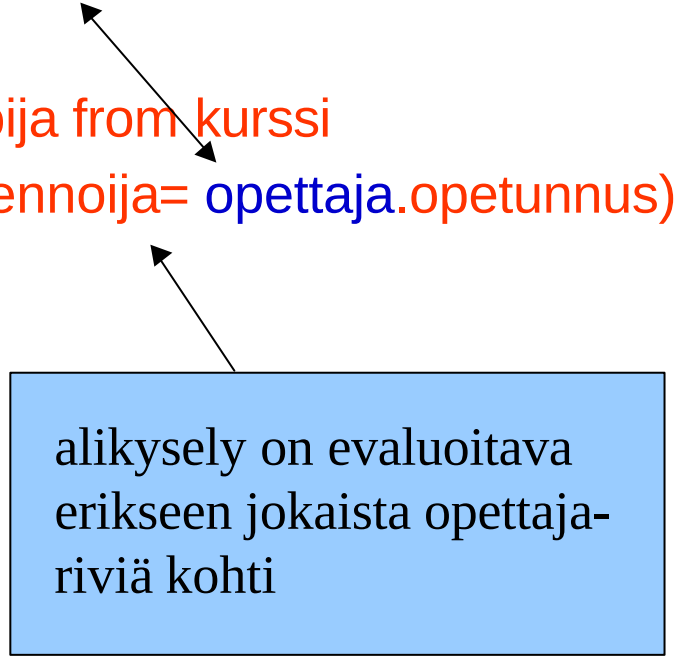
```
select nimi from opettaja
where opetunnus not in
   (select luennoija from kurssi)
order by nimi;
```



Alikyselyt

- Luennoivat opettajat, kytketyllä alikyselyllä

```
select nimi from opettaja
where
exists  (select luennoija from kurssi
                where luennoija= opettaja.opetunnus)
order by nimi;
```



Alikyselyt

- Vuoden 92 standardissa from-osaan sallittiin normaalien taulujen myös alikyselyiden tulostaulut

- from (alikysely) [[as] alias [(sarakeluettelo)]]
  - alikysely on normaali kysely
  - sarakeluettelo uudelleennimeää alikyselyn tulossarakkeet
  - tästä rakenteesta on hyötyä, jos halutaan yhdistää yksityiskohtaista tietoa ja yhteenvetotietoa tai eri perustein laskettuja yhteenvetotietoja samalle riville. Yksityiskohtien kyselyssä rakennetta ei tarvita.
  - yleensä rakenteen käyttö vain sotkee asioita - VÄLTÄ

Alikyselyt

- Esimerkki: Opiskelija(hetu, nimi, opintoviikot, …)

```
Select nimi, opintoviikot
from opiskelija,
        (select avg(opintoviikot) kaovt
        from opiskelija) as m
where oviikot > m.kaovt;
```

Huom. ei liitosehtoa joten tehdään ristitulo eli jokaiseen opiskelijariviin liitetään opintoviikkojen keskiarvo

Kysely tuottaa yhden rivin, jossa vain yksi sarake kaovt, opintoviikkojen keskiarvo

Yhteenvetokyselyt

- 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

Yhteenvetokyselyt

- Opiskelijoiden lukumäärä:
  - select count(*) from opiskelija;
- Count:n argumenttina 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;

Yhteenvetokyselyt

- Milloin pisimpään opiskelut helsinkiläinen opiskelija on aloittanut opintonsa?

  select min(aloitusvuosi) from opiskelija

  where kaupunki='Helsinki';

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

Yhteenvetokyselyt

- Yhteenvedon laskenta voidaan rajata vain keskenään erilaisiin arvoihin (projektioon). Tällainen rajaus on yleensä järkevä vain lukumääriä laskettaessa
- Monellako eri paikkakunnalla opiskelijat asuvat?
  - select count(distinct kaupunki) from opiskelija;

Yhteenvetokyselyt

- Yhteenvetofunktion sisältävään kyselyyn ei voi ottaa mukaan dataa niiltä yksittäisiltä riveiltä, joilta funktio lasketaan
- Minkä kurssien opintoviikkomäärä on suurin?
- EI siis ole mahdollista:

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

yksittäisen rivin dataa
miltä riviltä nimi poimittaisiin?

Yhteenvetokyselyt

- Minkä kurssin opintoviikkomäärä on suurin? Toimivia vaihtoehtoja:

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

Select nimi, opintoviikot from kurssi
where opintoviikot=
(select max(opintoviikot) from kurssi);

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

Yhteenvetokyselyt -ryhmät

- Jos kyselyyn liitetään ryhmittelymääre (Group by) muuttuu tuolosrivien muodostusperiaate jälleen:
  - **muodostetaan yksi tulosrivi kutakin ryhmää kohti**
  - group by -määreessä luetellaan sarakkeet, joiden arvojen 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.

Yhteenvetokyselyt -ryhmät

Group by A

Taulu X

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

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

A	B
•	10
•	7
•	3

Yhteenvetokyselyt -ryhmät

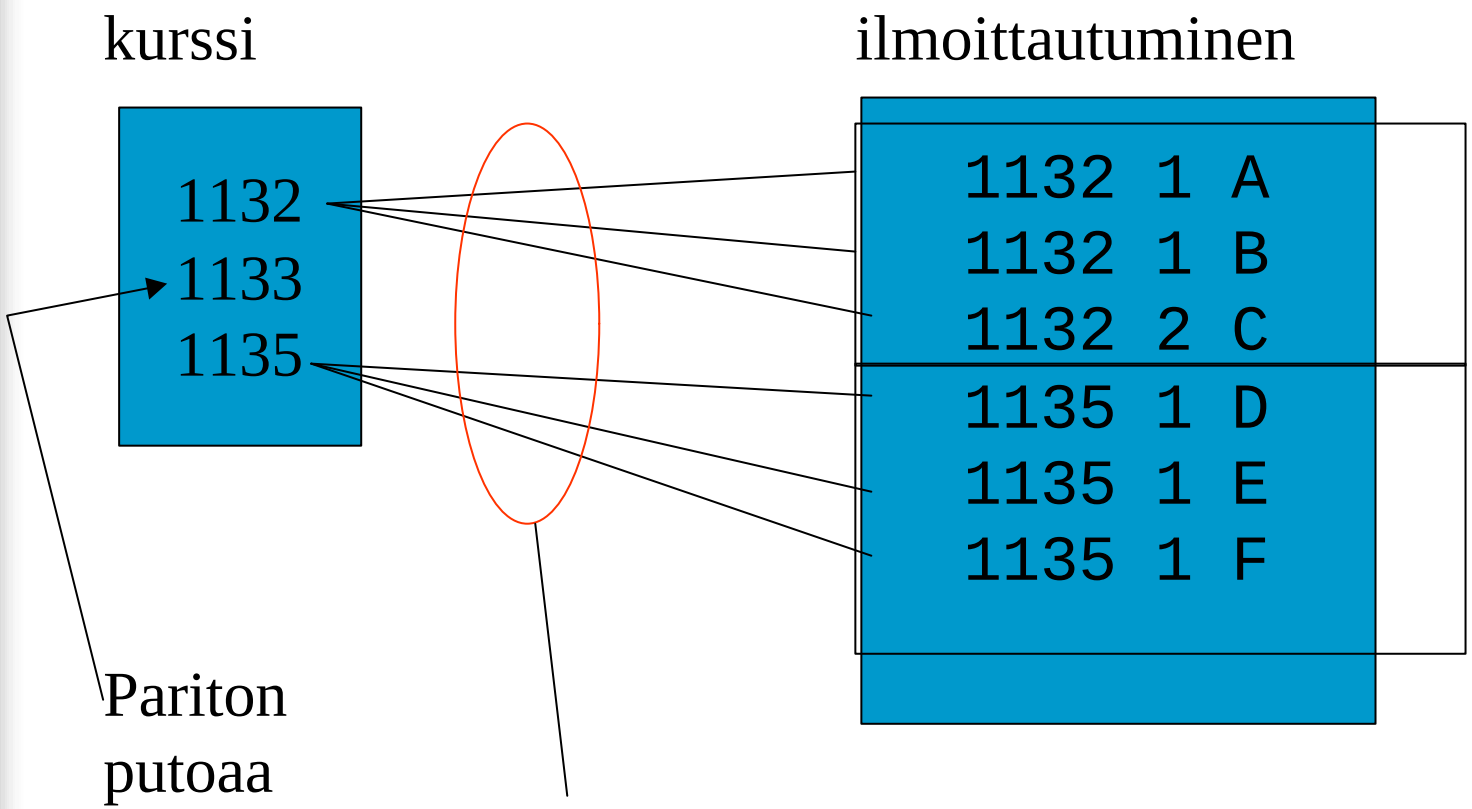
- Group by -lausetta käytettäessä tulostietoluettelossa voi olla yhteenvetofunktioiden lisäksi vain niitä sarakkeita,jotka esiintyvät group by -lauseessa.
- Kaikkien ryhmittelyyn käytettyjen sarakkeiden ei tarvitse olla mukana, mutta yleensä ne ovat

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

Ei anna tyhjien ryhmien ilmoittautujamäärää

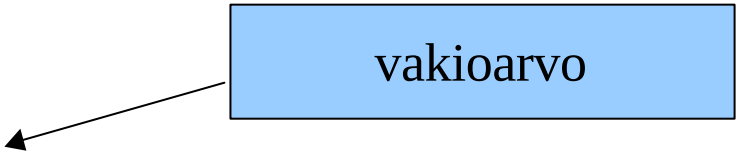
koska **nimi** on tulostietolistalla sen pitää olla myös ryhmittelytekijänä vaikka ilman sitä saataisiin samat ryhmät

Yhteenvetokyselyt -ryhmät



Yhteenvetokyselyt -ryhmät

```
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));
```



Yhteenvetokyselyt -ryhmät

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

Yhteenvetokyselyt

- Yhteenvetofunktioilla voi laskea, mutta niitä ei voi ketjuttaa (eli toinen on toisen argumenttina)
- Millä kurseilla on suurin keskimääräinen ryhmäkoko, ei onnistu seuraavasti

```
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

Yhteenvetokyselyt -ryhmät

```
select koodi, nimi, opiskelijoita/ryhmia
   from kurssi,
  (select kurssikoodi, count(*) opiskelijoita
      from ilmoittautuminen
      group by kurssikoodi) as ilm,
  (select  kurssikoodi, count(*) ryhmia
      from harjoitusryhma
      group by kurssikoodi) as ryhm
  where kurssi.koodi= ilm.kurssikoodi and
      kurssi.koodi= ryhm.kurssikoodi and
      opiskelijoita/ryhmia = (XXXXX –seuraavalla kalvolla)
```

Yhteenvetokyselyt -ryhmät

- XXXXX=

```
select max(opiskelijoita/ryhmia)
  from
 (select kurssikoodi, count(*) opiskelijoita
    from ilmoittautuminen
    group by kurssikoodi) as i,
 (select  kurssikoodi, count(*) ryhmia
    from harjoitusryhma
    group by kurssikoodi) as ry
 where i.kurssikoodi = ry.kurssikoodi
```

Näkymät

Perustauluja ja johdettuja tauluja.

Johdettu taulu eli näkymä (view) määritellään kyselyn avulla

CREATE VIEW taulunimi

[(sarake1,sarake2,…)] AS kysely;

Create view miesopiskelijat (hetu, nimi, ovt) as
(select hetu, nimi, oviikot
from opiskelija
where mod((substr(hetu,8,3),2)=1);

111170-123X

Näkymät

- Näkymää käytetään kuten muitakin tauluja; lasketaan kuinka monta ilmoittautumista miespuoliset opiskelijat ovat tehneet syksyllä 2003

```
Select count(*)
From ilmoittautuminen i, miesopiskelijat m
where i.hetu=m.hetu and
        lukuvuosi=2003 and lukukausi='S'
```

Todellisuudessa näkymään liittyvä kysely suoritetaan aina kun näkymää käytetään!

Miksi näkymiä tarvitaan?

- Tietoriippumattomuus
  - esim.
    Create view ilmo as
    select kurssikoodi, ryhmanro, opisknro, ilm_aika
    from ilmoittautuminen
    union
    select kurssikoodi, ryhmannro, opisknro, ilm_aika
    from vanhat_ilmoittautumiset;
- Täsmäsuojaus
  - esim. create view oma_kurssi as
    select * from kurssi where luennoija=user;
- Kyselyiden helpottaminen

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),
        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 (kaikki kyselyn tulosrivit lisätään):

```
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

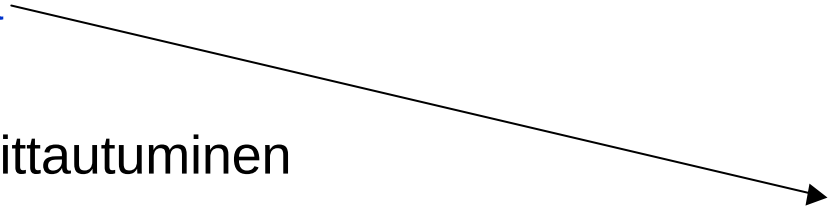
SQL - Tietokannan ylläpito

- 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 sarakkeiden sisältöä,
  - muutetaan kaikki where-ehdon täyttävät 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

- 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 - 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]**
- Tapahtuma voidaan päättää myös perumalla sen aikaansaamat muutokset komennolla **rollback [work]**
- Tilisiirtotapahtuma olisi kokonaisuudessaan 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

SQL - Tietokantatapahtuma (transaktio)

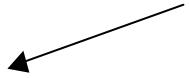
- Yritetään poistaa tyhjät harjoitusryhmät, oletetaan, että ilmoittautumisten viiteavaimeen liittyy on delete cascade -määre

```
commit;
select count(*) from ilmoittautumiset;
>> 3500 <<
```

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 <<<
```



Ei ole ihan oikein

(oho!)

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 <<<
rollback;
select count(*) from ilmoittautumiset;
>> 3500 <<
```

Ei ole ihan oikein

(oho!)

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 <<<
rollback;
select count(*) from ilmoittautumiset;
>> 3500 <<
```

Ei ole ihan oikein

(oho!)