SQL -valintaehto

```
CREATE TABLE opettaja (
    opetunnus varchar(12) NOT NULL ,
    nimi varchar(40) NOT NULL ,
    puhelin varchar(12) ,
    tyohuone varchar(12),
PRIMARY KEY (opetunnus) ) ;
CREATE TABLE opiskelija (
   onumero numeric(5) NOT NULL ,
   nimi varchar(40) NOT NULL ,
   paa_aine varchar(12) NOT NULL,
   kaupunki varchar(30),
   aloitusvuosi numeric(4),
PRIMARY KEY (onumero ) ) ;
```

SQL-valintaehto

- Opettajien nimet:
  - select nimi from opettaja;
- Opiskelijoiden pääaineet
  - select distinct paa_aine from opiskelija;
- Tietojenkäsittelytieteen pääaineopiskelijoiden nimet
  - select nimi from opiskelija where paa_aine ='TKT';

SQL-valintaehto

- Espoossa asuvat matematiikan opiskelijat

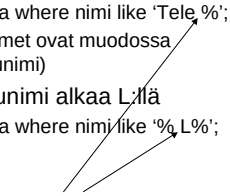
  select * from opiskelija where paa_aine='MAT' and kaupunki='Espoo';

- Vuosina 1994-1997 opintonsa aloittaneet opiskelijat:

```
select * from opiskelija
where aloitusvuosi >=1994 and aloitusvuosi<=1997;  tai
select * from opiskelija
where aloitusvuosi between 1994 and 1997;    tai
select * from opiskelija
where aloitusvuosi in (1994,1995,1996,1997);
```

SQL-valintaehto

- Opiskelijat joiden sukunimi on Tele
  - select * from opiskelija where nimi like 'Tele %';
  - (esimerkkitaulussa nimet ovat muodossa sukunimi+'space'+etunimi)
- Opiskelijat, joiden etunimi alkaa L:llä
  - select * from opiskelija where nimi like '% L%';



SQL-funktioita

- Opettajien sukunimet (Oraclessa)

  select substr(nimi, 1, instr(nimi,' ')-1) from opettaja;
- substr(sarake, alku,pituus) eristää osamerkkijonon
  - standardissa substring(sarake FROM alku FOR pituus)
  - Solidissa substring(sarake, alku, pituus)
- instr(sarake,merkkijono) antaa upotetun merkkijonon alkukohdan tai 0, jos ei löydy
  - std:ssa ja Solisissa: position(merkkijono IN sarake)

SQL-funktioita

- Opettajat, joiden puhelinnumero on parillinen

```
select * from opettaja
where substr(puh,length(puh),1) in
('0','2','4','6','8');
 tai
select * from opettaja
where mod(to_number(puh),2)=0;
```

SQL- tuloksen järjestäminen

- Järjestetty tulos on käyttäjän kannalta helpommin hahmotettavissa, soveltuu tiedon etsintään
- SQL:ssä järjestys saadaan aikaan järjestysmääreellä
  - order by sarake [{asc[ending] | desc[ending] }] [, …]
  - ascending = nouseva järjestys, pienemmät arvot alussa (oletusjärjestys ellei mainita)
  - descending = laskeva järjestys

SQL- tuloksen järjestäminen

- Jos järjestysmääressä annetaan useita sarakkeita
  - järjestetään rivit ensin ensimmäisenä annetun sarakkeen perusteella
  - seuraavaksi ne rivit, joilla ensimmäisenä annetussa sarakkeessa on sama arvo järjestetään luettelon 2. sarakkeen perusteella
  - seuraavaksi ...

Järjestys

	Order by A	Order by A,B	Order by A desc, B
A B C D	A B C D	A B C D	
2 4 8 7	1 4 6 7	1 1 4 2	A B C D
1 4 6 7	1 1 4 2	1 4 6 7	3 1 5 2
3 1 5 2	1 5 5 2	1 5 5 2	3 2 4 6
2 3 5 1	2 4 8 7	2 3 5 1	2 3 5 1
1 1 4 2	2 3 5 1	2 4 8 7	2 4 8 7
3 2 4 6	3 1 5 2	3 1 5 2	1 1 4 2
1 5 5 2	3 2 4 6	3 2 4 6	1 4 6 7
			1 5 5 2

SQL -järjestäminen

- Merkkitiedon järjestäminen saattaa olla järjestelmän asetuksista riippuvaa ja toimia eri järjestelmissä eri tavoin
  - Esim. HY:n Oracle järjestää suomenkielisen aakkosjärjestyksen mukaisesti (V=W) eikä järjestäessään erottele isoja ja pieniä kirjaimia
  - Usein järjestäminen tapahtuu kuitenkin merkin merkkikoodin mukaisesti

Järjestäminen

- Opiskelijat pääaineittain opiskeluajan perusteella järjestettynä vanhimmasta nuorimpaan

```
select paa_aine, aloitusvuosi, nimi
from opiskelija
order by paa_aine, aloitusvuosi;
```

- Järjestyksen määräämiseen käytettävien sarakkeiden on syytä olla tulostietolistan alussa (käyttöliittymätekniset syyt, ei SQL:n vaatimus)

SQL-Liitokset

- Jos kyselyn from osassa on useampia tauluja, muodostetaan näiden taulujen ristitulo, ellei where-osassa ole ehtoa, joka kytkisi rivit yhteen
- Jos where-osassa on rivit yhteenkytkevä ehto on kyseessä liitosoperaatio

SQL-Liitokset

```
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)
```

- Kuka luennoi mitäkin kurssia:

```
select  kurssi.nimi, opettaja.nimi
    from opettaja, kurssi
    where luennoija=opetunnus
    order by kurssi.nimi;
```

SQL-Liitokset

- Liitos tehdään useimmiten vertaamalla taulun avainta toisessa taulussa olevaan siihen viittaavaan viiteavaimeen.
- Muunkinlaiset liitokset toki mahdollisia:
- Opiskelijat, jotka toimivat ehkä myös opettajina

```
select opiskelija.nimi
    from opiskelija, opettaja
    where opiskelija.nimi=opettaja.nimi
    order by opiskelija.nimi;
```

SQL-Liitokset

- Taulujen järjestyksellä from osassa ei ole merkitystä kyselyn vastauksen kannalta
  - sillä saattaa olla merkitystä kyselyn suoritusajan kannalta, mutta tämä on järjestelmäkohtaista
- Ehtojen järjestyksellä where-osassa ei ole merkitystä kyselyn vastauksen kannalta
  - silläkin saattaa olla merkitystä kyselyn suoritusajan kannalta, mutta tämä on järjestelmäkohtaista

SQL-Liitokset

- 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

SQL-Liitokset

- 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

SQL-Liitokset

- Vuoden 92 standardissa from-osaan sallittiin normaalien taulujen lisäksi myös kyselyiden tulostaulut ja liitostulokset (ei Oracle)
- from (alikysely) [[as] alias [(sarakeluettelo)]]
  - alikysely on normaali kysely
  - sarakeluettelo uudelleennimeää alikyselyn tulossarakkeet
  - tästä rakenteesta on hyötyä, jos halutaan yhdistää yksityiskohtaista tietoa ja yhteenvetotietoa, esimerkki myöhemmin yhteenvetotietojen yhteydessä
  - yleensä rakenteen käyttö vain sotkee asioita - VÄLTÄ

SQL-Liitokset

- Liitostulokset from osassa:
- from … taulu1 [<join type>]JOIN taulu2 [ON <liitosehdot>]
- <join type> = inner | left outer | right outer | full outer
- Inner join on normaaliliitos (oletusarvo, jos join type-määrettä ei anneta)

```
select  kurssi.nimi, opettaja.nimi
    from opettaja join kurssi on luennoija=opetunnus
    order by kurssi.nimi
```

- Tämä on siis toinen tapa esittää liitos
- Jos kurssilla ei ole opettajaa, ei tulokseen tule mittään tietoa kurssista

SQL-Liitokset

- Ulkoliitokset
  - ulkoliitoksella saadaan parittomaksi normaali-liitoksessa jäävät rivit mukaan tulokseen
- Kaikki kurssit, luennoijatietoineen:

```
select kurssi.nimi, opettaja.nimi
    from kurssi left outer join luennoija on luennoija=opetunnus
    order by kurssi.nimi
```

  - nyt luennoijattomatkin kurssit tulevat mukaan, luennoija.nimi saa arvon null

```
select opettaja.nimi, kurssi.nimi
from kurssi right outer join luennoija on luennoija=opetunnus
    order by  opettaja.nimi
```

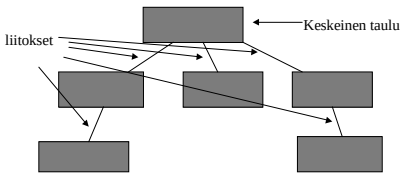
- näin saadaan kaikki opettajat, myös ne, jotka eivät luennoi

SQL-Liitokset

- Tyypillinen virhe liitoksissa on jättää jokin liitosehto pois, jolloin tuloksen rivijoukko tulee huomattavasti suuremmaksi kuin pitäisi
- jos from osassa on n kpl liitettäviä tauluja tarvitaan vähintään n-1 liitosehtoa. Taulujen liittäminen voi perustua useaan sarakkeeseen, jolloin ehtolausekkessa tarvittavien alkeisehtojen määrä voi moninkertaistua.

SQL-Liitokset

- Yleensä kyselyt rakentuvat siten, että niissä on jokin keskeinen taulu johon, muita liitetään. Voi olla ettei tuosta keskeisestä taulusta tule mitään dataa tulokseksi.



SQL - liitokset

```
CREATE TABLE harjoitusryhma (
    kurssikoodi  numeric(4) NOT NULL ,
    ryhmanro  numeric(2) NOT NULL ,
    viikonpaiva  varchar(12) NOT NULL ,
    alkamisaika  numeric(2) NOT NULL ,
    sali varchar(12) NOT NULL,
    opettaja varchar(12) NOT NULL,
    PRIMARY KEY (kurssikoodi, ryhmanro) ,
    FOREIGN KEY (kurssikoodi) REFERENCES kurssi,
    FOREIGN KEY (opettaja) REFERENCES opettaja )
;
```

SQL - liitokset

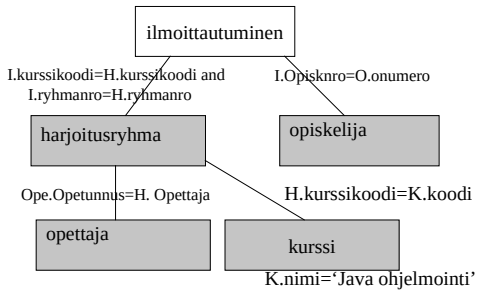
```
CREATE TABLE ilmoittautuminen (
    kurssikoodi numeric(8) not null,
    ryhmanro numeric(2) not null,
    opisknro  numeric(5) NOT NULL ,
    ilm_aika  date NOT NULL ,
    PRIMARY KEY (opisknro, kurssikoodi) ,
    FOREIGN KEY (kurssikoodi, ryhmanro)  REFERENCES
        harjoitusryhma ,
    FOREIGN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

SQL - liitokset

- Raportti kurssin ‘Java ohjelmointi’ harjoitusryhmistä

```
select H.ryhmanro rno, Ope.nimi ope, H.viikonpaiva,
    H.alkamisaika, O. Nimi opiskelija
from Harjoitusryhma H, opettaja Ope, opiskelija O,
    ilmoittautuminen I, kurssi K
where
    H.kurssikoodi=K.koodi and
    I.kurssikoodi=H.kurssikoodi and
    I.ryhmanro=H.ryhmanro and
    Ope.Opetunnus=H. Opettaja and
    I.Opisknro=O.onumero and K.nimi=‘Java ohjelmointi’
order by H.ryhmanro, O.nimi;
```

SQL - liitokset



SQL - alikyselyt

- Alikyselyllä tarkoitetaan kyselyyn upotettua toista kyselyä. Upotettua kyselyä voidaan käyttää kyselyn from osassa mutta myös where osassa valintaehtojen operandina.
- On pidettävä mielessä, että alikyselykin tuottaa tuloksenaan taulun
- Alikyselyiden käyttöön valintaehdoissa on omia predikaatteja ja lisätarkenteita, jotka määrittelevät, miten ehdon operandia sovelletaan alikyselyn tulokseen

SQL - alikyselyt

- Vertailuoperaatio IN
  - vertailuoperaatiossa a IN B operandi B on joukko, joten sen tilalle alikysely soveltuu suoraan
  - a in (alikysely) on tosi, jos vakio a tai sarakkeen a arvo sisältyy alikyselyn tulokseen.
  - Luennoivat opettajat:

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

SQL - alikyselyt

  - Luennoimattomat opettajat:

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

- Standardin mukaan ja Oraclessa IN vertailun osapuolena voi olla myös sarakeyhdistelmä (Solidissa vain yksittäinen sarake).

SQL - alikyselyt

- Harjoitusryhmät, joihin ei ole ilmoittautunut ketään

```
select nimi, ryhmanro
    from kurssi, harjoitusryhma
    where kurssi.koodi=harjoitusryhma.kurssikoodi and
    (kurssikoodi, ryhmanro) not in
        (select kurssikoodi, ryhmanro from ilmoittautuminen)
order by nimi, ryhmanro;
```

SQL - alikyselyt

- Muut vertailuoperaatiot ovat yksittäisten arvojen välisiä. Tällaisten osapuolina voi käyttää alikyselyä, jos on varmaa että kysely tuottaa tuloksenaan enintään yhden rivin. Tulee ajoaikainen virhe, jos ei tuotakaan. Tällaisia ovat tyypillisesti yhteenvetokyselyt (myöhemmin)
- Lisämääreillä SOME = ANY ja ALL voidaan yksittäisten arvojen vertailuun perustuvaa operaatiota soveltaa arvojoukoon:
  - SOME = jokin joukon arvo toteuttaa ehdon
  - ALL = kaikki joukon arvot toteuttavat ehdon

SQL - alikyselyt

- Luennoivat opettajat:

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

- ‘ = SOME’ on siis sama kuin ‘IN’
- Mistä kurssista saa eniten opintoviikkoja?

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

SQL - alikyselyt

- Yksittäisen arvon tyhjyyttä testataan IS NULL tai IS NOT NULL predikaateilla
- Alikyselyn tyhjyyden testaamiseen ovat tarjolla EXISTS ja NOT EXISTS predikaatit
  - Exists (alikysely) on tosi, jos alikysely tuottaa ainakin yhden tulosrivin
  - Not exists (alikysely) on tosi, jos vastaus on tyhjä eli ei yhtään riviä (ei tarjolla Solidissa)
- Kyselyn tyhjyyttä on mielekästä testata vain ns. **kytkettyjen alikyselyjen** yhteydessä

SQL - alikyselyt

- Aiemmissa esimerkeissä alikysely on ollut täysin riippumaton sen sisältävästä pääkyselystä (= sen voisi suorittaa erillisenä kyselynä ja sen tulos olisi sama kuin pääkyselyyn upotettuna = se ei viittaa mihinkään pääkyselyn elementtiin)
- Kytketyssä alikyselyssä tilanne on toinen:
- sitä ei voi suorittaa erillisenä, koska jokin siinä oleva valintaehto tai tulostiedon määritys käyttää hyväkseen pääkyselyn saraketta

SQL - alikyselyt

- Luennoivat opettajat:

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

- Tässä alikysely suoritetaan jokaista opettaja-riviä kohden erikseen ja opettajarivi tulee tulokseen, jos alikyselyn tulos ei ole tyhjä.

SQL - alikyselyt

Luennoijat jotka pitävät harjoituksia kursseillaan

```
Select O.nimi from opettaja O, kurssi K
   where O.opetunnus=K.luennoija and
         O.opetunnus in (select opettaja from
            harjoitusryhma where kurssikoodi=K.Koodi);
```

```
Select O.nimi from opettaja O, kurssi K
   where O.opetunnus=K.luennoija and
   exists (select ‘A’ from harjoitusryhma
          where opettaja=o.opetunnus and kurssikoodi=K.Koodi);
```

```
Select O.nimi from opettaja O, kurssi K
where O.opetunnus=K.luennoija and
(O.opetunnus,K.koodi) in
     (select opettaja, kurssikoodi from harjoitusryhma );
```

SQL - alikyselyt

- Ketjutus

```
select nimi from opettaja
where opetunnus in
    (select opettaja from harjoitusryhma
     where kurssikoodi in
         (select koodi from kurssi
         where nimi like 'Java%'));
```

Java-alkuisten kurssien opettajien nimet.

SQL - joukko-opin perusoperaatiot

- Kyselyjä voi yhdistää union (yhdiste), intersect (leikkaus) ja except (erotus) operaatioilla. Järjestysmääre koskee koko tulosta.

```
Select ……
  [union | intersect | except]
  kysely
order by ......;
```

  - union karsii toistuvat rivit
  - Oraclessa except:n tilalla minus

SQL - joukko-opin perusoperaatiot

- Opiskelijat, jotka ovat myös opettajia

```
select nimi from opettaja
intersect
select nimi from opiskelija
order by nimi;
```