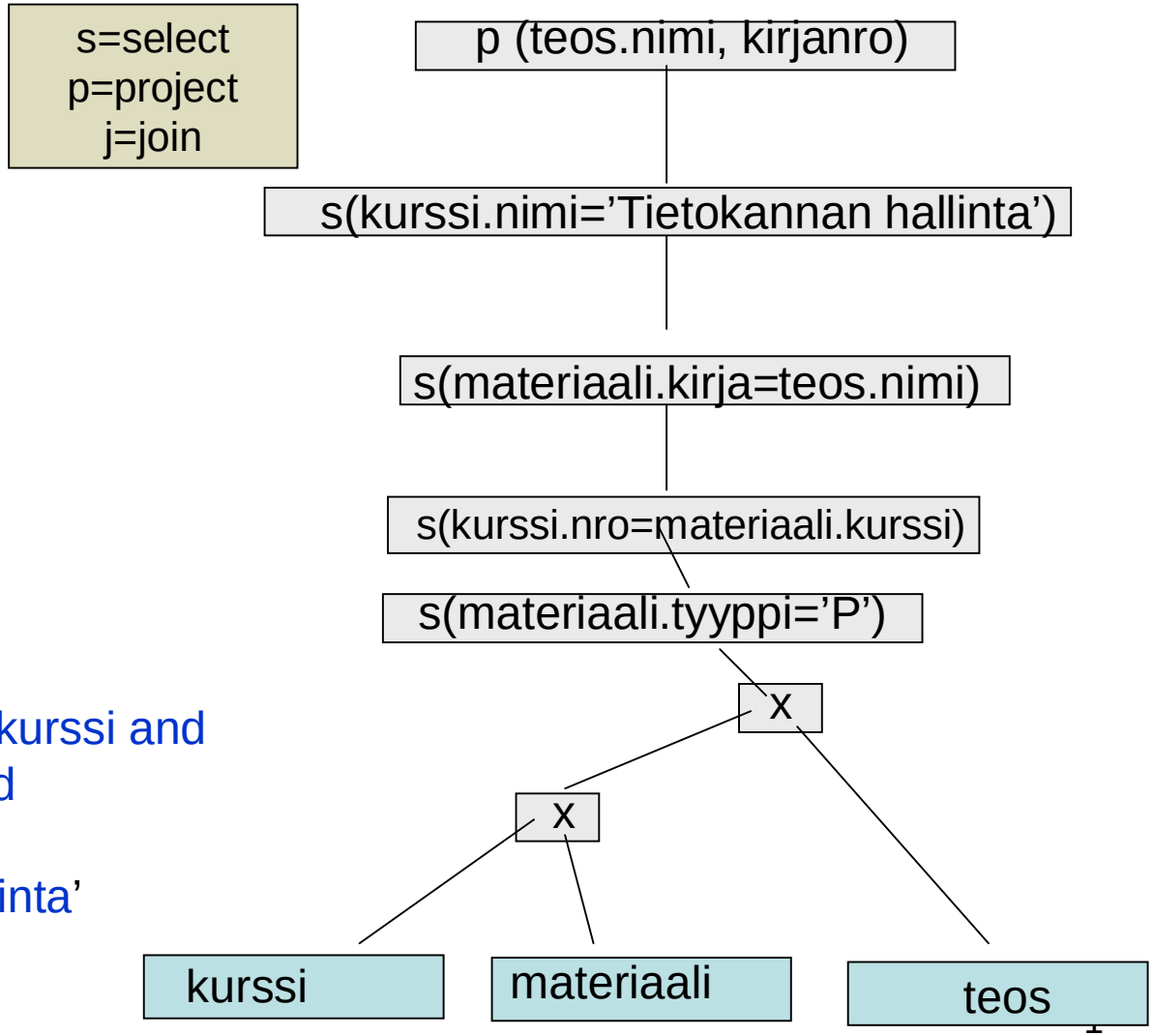
Kyselypuut ja ekvivalenssi

Sisäisessä esityksessä kyselyt esitetään kyselypuuna

- lehdissä taulut
- juuressa lopputulos
- välisolmuina suoritettavat operaatiot

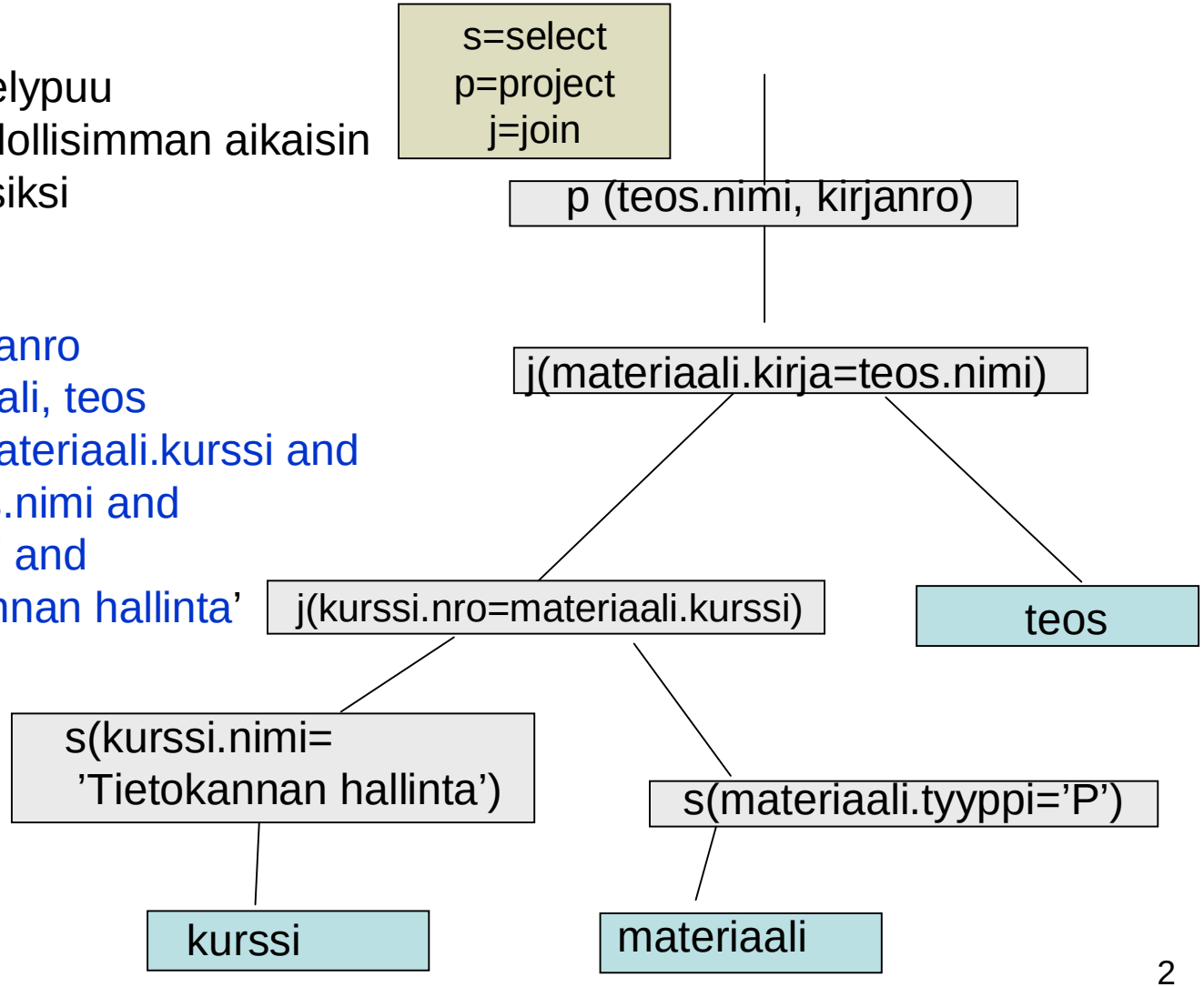
```
select teos.nimi, kirjanro
from kurssi, materiaali, teos
where kurssi.nro=materiaali.kurssi and
materiaali.kirja=teos.nimi and
materiaali.tyyppi='P' and
kurssinimi='Tietokannan hallinta'
```

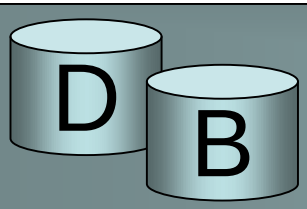


Kyselypuut ja ekvivalenssi

Optimoitu kyselypuu
- valinnat mahdollisimman aikaisin
- ristitulot liitoksiksi

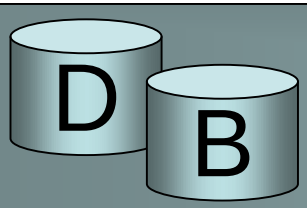
```
select teos.nimi, kirjanro
from kurssi, materiaali, teos
where kurssi.nro=materiaali.kurssi and
materiaali.kirja=teos.nimi and
materiaali.tyyppi='P' and
kurssinimi='Tietokannan hallinta'
```





Kyselypuut ja ekvivalenssi

- Kyselypuun muokkauksessa vaihdetaan operaatioiden järjestyksiä tai korvataan jokin operaatio toisella operaatiolla
- Muokatun kyselypuun pitää tuottaa sama tulos kuin aiemman
- Kaksi operaatiosarjaa ovat ekvivalentteja, jos ne tuottavat saman tuloksen lähtötaulujen kaikilla instansseilla
  - ekvivalenssisäännöt määrittelevät millaiset muunnokset ovat sallittuja ilman, että tulos muuttuu



Kyselypuut ja ekvivalenssi

- Valinnat voidaan yhdistää tai purkaa sisäkkäisiksi

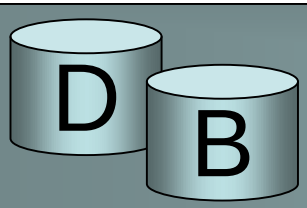
  $\sigma_{c1 \wedge c2 \wedge \ldots \wedge ck}(R) \equiv \sigma_{c1}(\sigma_{c2}(\ldots \sigma_{ck}(R) \ldots))$

- Valinnat ovat vaihdannaisia (commutative)

  $\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R))$

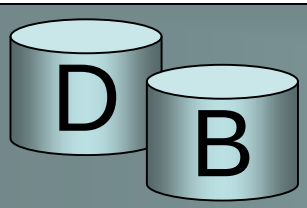
- Peräkkäisten projektioiden lopputuloksen määrä viimeisin projektio

- $\pi_{a1}(R) \equiv \pi_{a1}(\pi_{a2}(\ldots(R)))$



Kyselypuut ja ekvivalenssi

- Ristitulo ja liitos ovat kommutatiivisia, osapuolten järjestys voidaan vaihtaa (kumpi on ulompi)
  - RxS ≡ SxR
  - R⋈S ≡ S⋈R
- Ristulo ja liitos ovat assosiatiiviset (suoritusjärjestys on vaidettavissa)
  - Rx(SxT) ≡(RxS)xT

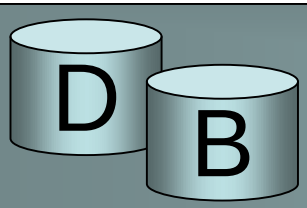


Kyselypuut ja ekvivalenssi

- Projektion ja valinnan järjestys on vaihdettavissa kunhan kaikki valinnassa esiintyvät sarakkeet ovat mukana projektiossa

  $\sigma_c(\pi_a(R)) \equiv \pi_a(\sigma_c(..(R)))$
- Valinta ja ristitulo voidaan yhdistää liitokseksi, jos kyseessä on liitosehto

  $R \bowtie_c S \equiv \sigma_c(RxS)$
- Valinta voidaan siirtää tapahtuvaksi ennen ristituloa tai liitosta, mikäli se koskee vain näiden argumenttia
- $\sigma_c(RxS) \equiv \sigma_c(R)xS$



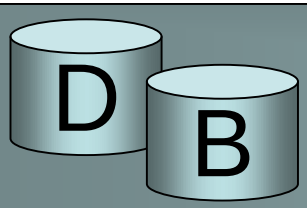
Kyselypuut ja ekvivalenssi

- Yleisesti valinta $\sigma_c(RxS)$ voidaan jakaa tauluja R ja S koskeviin osiin sekä liitosehtoon eli

  $\sigma_c(RxS) \equiv \sigma_c(\sigma_{cR}(R) \times \sigma_{cs}(S))$
- Projektio voidaan jakaa osiin jotka suorittetaan ennen ristituloa

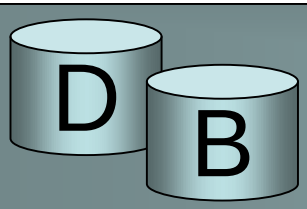
  $\pi_a(RxS) \equiv \pi_{a1}(R) \times \pi_{a2}(S)$
- Projektio voidaan jakaa osiin, jotka suoritetaan ennen liitosta, jos liitosehdon sarakkeet säilyvät

$\pi_a(R \bowtie S) \equiv \pi_{a1}(R) \bowtie \pi_{a2}(S)$



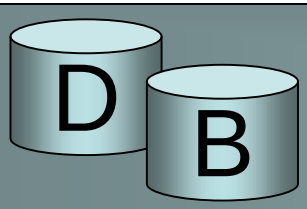
Heuristisia muokkaussääntöjä

- Konjunktiiviset valinnat (and-yhdistetyt) puretaan sarjaksi yksittäisiä valintoja
- Painetaan valinnat niin alas puussa kuin mahdollista
- Järjestellään lehtisolmut edulliseen järjestykseen
  - ensin mahdollisimman rajaavat valinnat
    = pieni rivijoukko odotettavissa tulokseksi
  - Yritetään välttää ristituloja
- Yhdistetään ristitulo ja seuraava valinta liitokseksi
- Yhdistetään operaatiot, jotka voidaan suorittaa yhdellä algoritmilla
- Vältellään välitulosten kirjoitusta



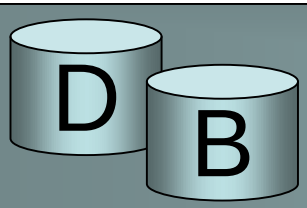
Rajaavuus

- Rajaavuutta (selectivity) mitataan rajaussuhteella (reduction factor):
- tuloksen rivimäärä / lähtöjoukon rivimäärä
- Rajaavuuden arviointi ilman tilastotietoja on hankalaa
  - Kun testataan avaimen yhtäsuuruutta suhteessa vakioon tuloksen koko on enintään 1 ja rajaussuhde 1/taulun koko (jos taulun kokoa ei tiedetä, sen tilalla voidaan käyttää taululle tehdyn tilanvarauksen kokoa, tai jotain vakiota, jos tätäkään ei tiedetä.)
  - sarakke=vakio valinta tuottaa tasaisen jakautuman oletuksella rajaussuhteen 1/sarakkeen arvojen lukumäärä. (jos sarakkeella on hakemisto voi arvojen lukumäärä olla tiedossa, tietotyypin määritys voi rajata arvojoukon esim. numeric(2) enintään 100 arvoa, järjestelmä voi käyttää oletusarvoa esim. 1/10, jos sillä ei ole parempaa tietoa)



Rajaavuus

- Täsmällisemmän tiedon saanti rajaavuudesta edellyttää tilastotietoa:
  - taulun koko (riveinä, sivuina)
  - sarakkeen arvojen lukumäärä (riittää, jos oletetaan tasainen jakautuma)
  - sarakkeen arvojakautuma (histogrammi) käyttökelpoinen, jos jakautuma on hyvin vino.

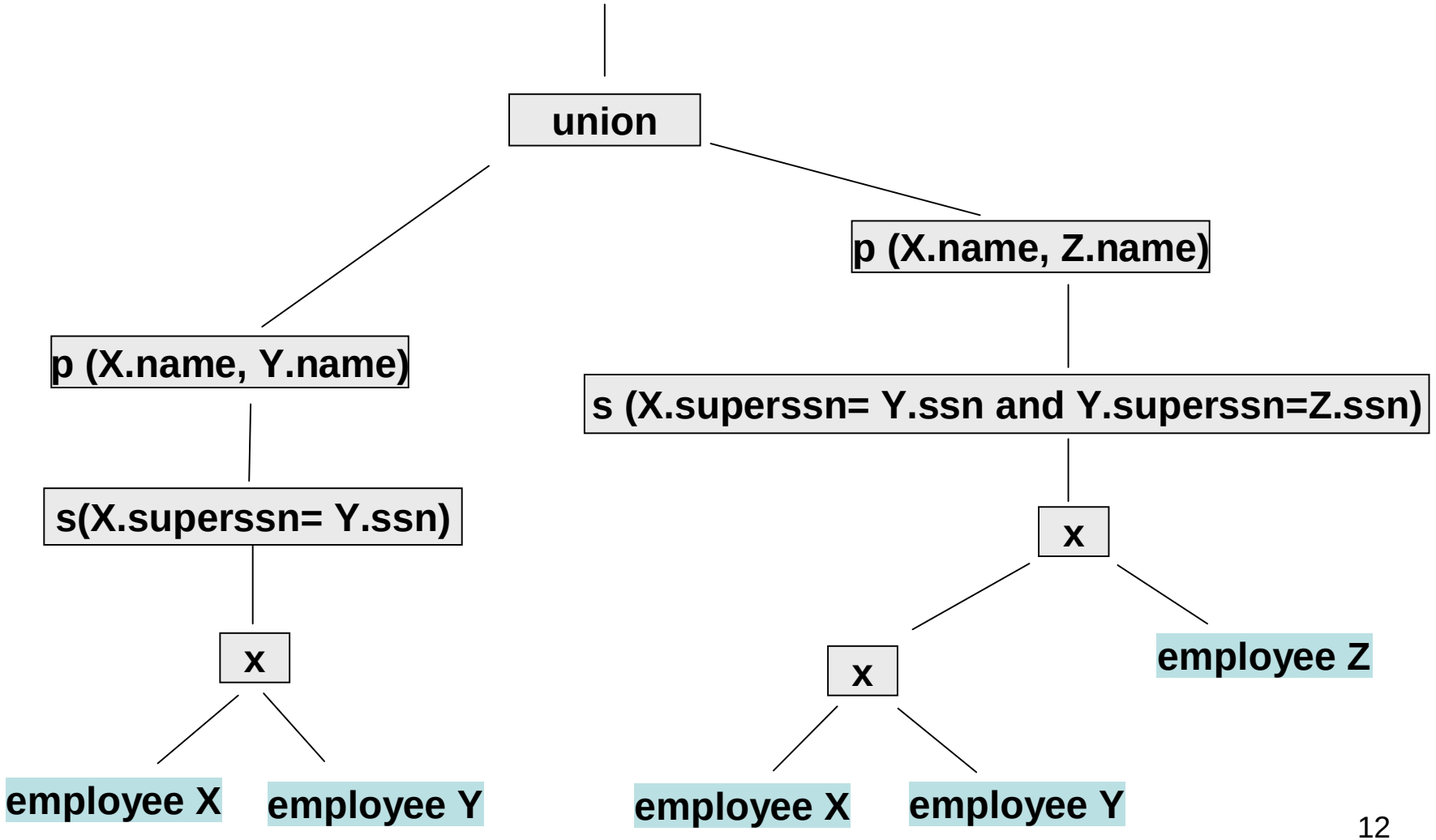


Esimerkki kyselypuusta

- Tutkitaan kyselyä

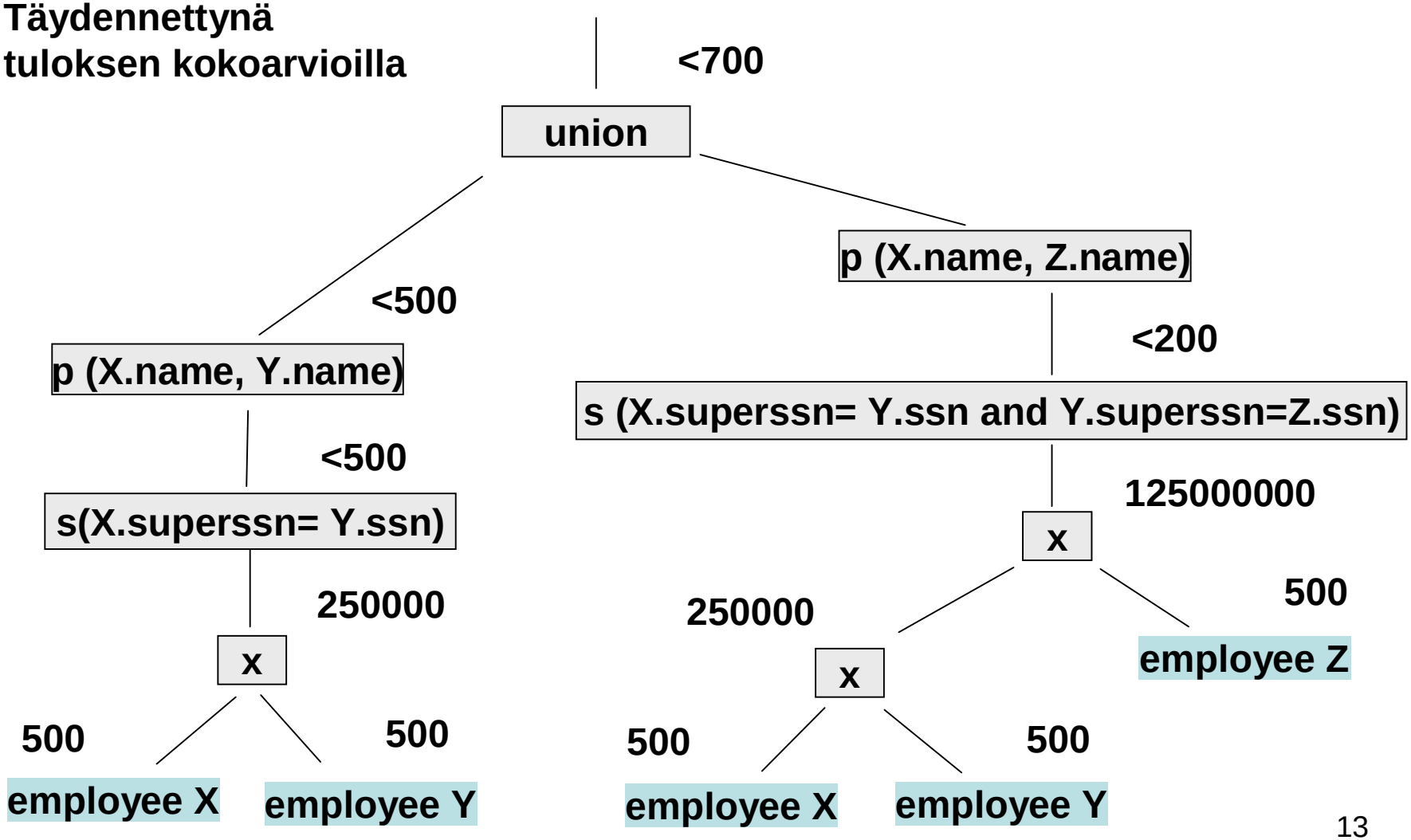
```
SELECT X.NAME, Y.NAME
FROM EMPLOYEE X, EMPLOYEE Y
WHERE X.SUPERSSN = Y.SSN
UNION
SELECT X.NAME,Z.NAME
FROM EMPLOYEE X, EMPLOYEE Y, EMPLOYEE Z
WHERE X.SUPERSSN = Y.SSN AND Y.SUPERSSN = Z.SSN
```

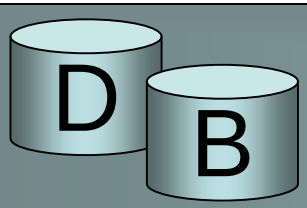
Esimerkki kyselypuusta



Esimerkki kyselypuusta

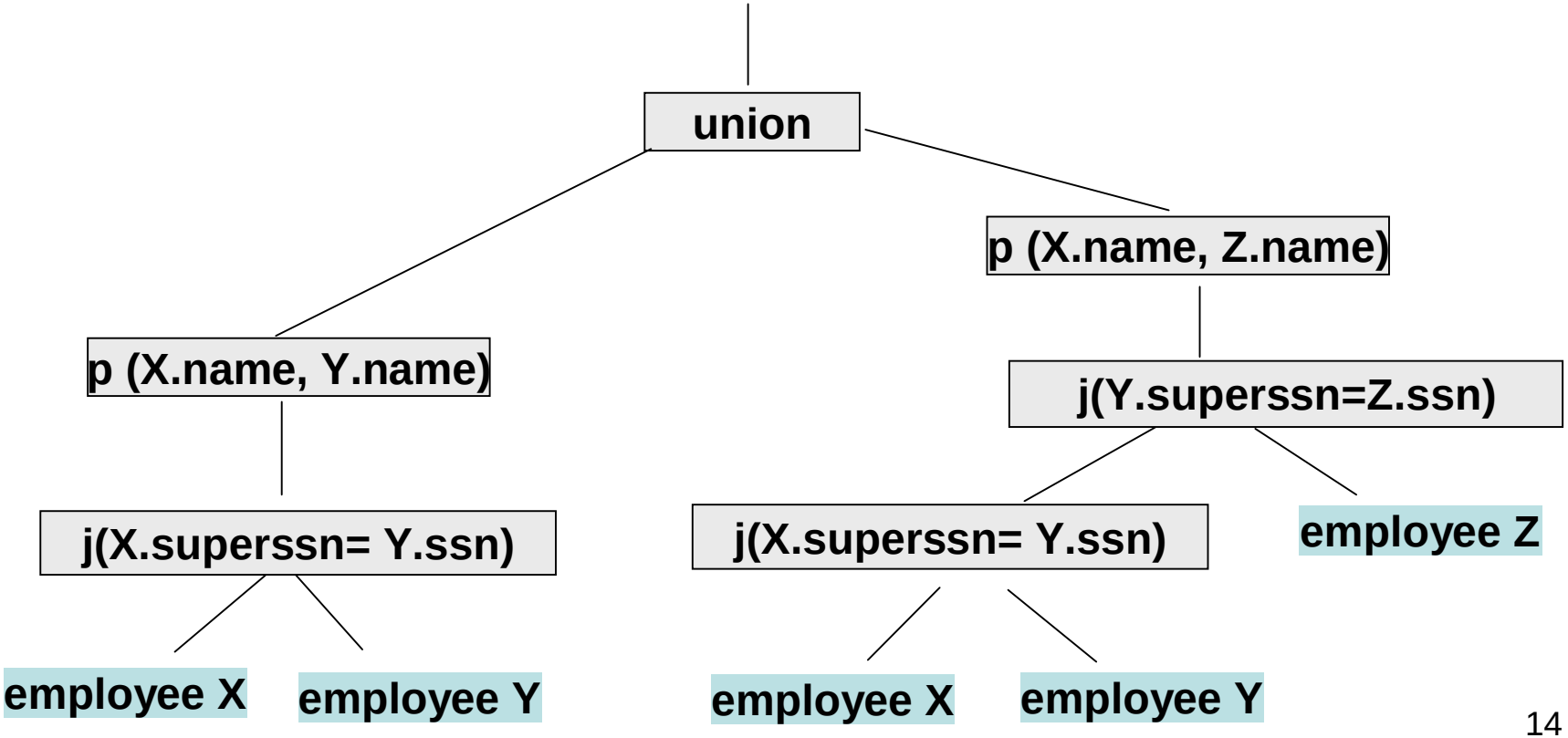
Täydennettynä tuloksen kokoarvioilla





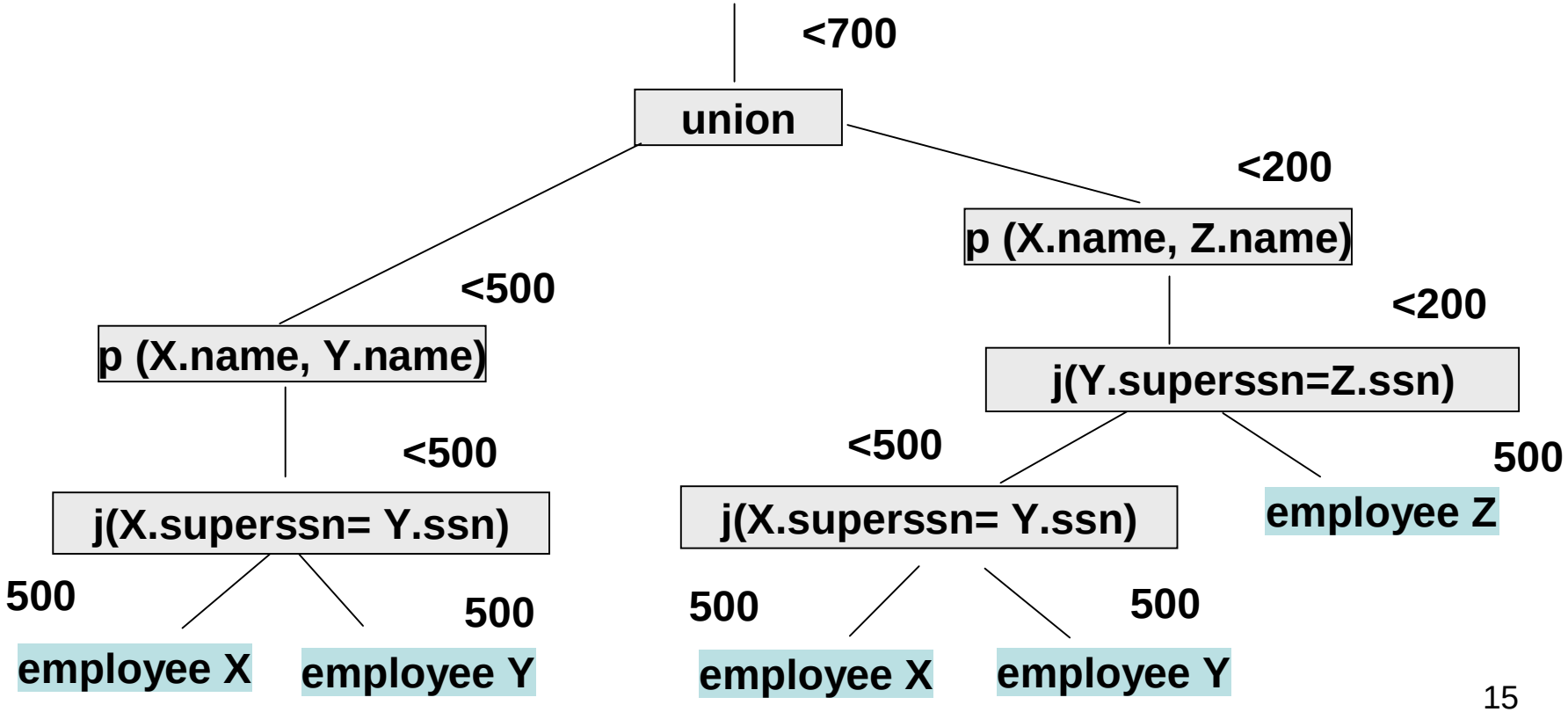
Esimerkki kyselypuusta

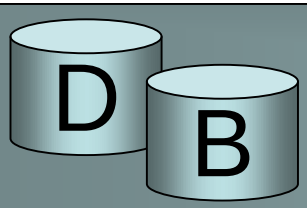
- Korvataan ristitulo ja siitä seuraava valinta liitoksilla



Esimerkki kyselypuusta

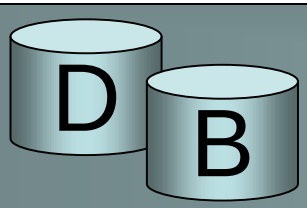
- täydennettynä kokoarvioilla





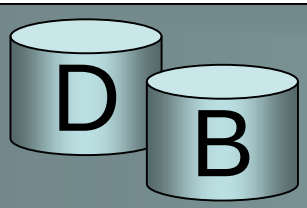
Kustannuslaskentaoptimointi

- Laskenta edellyttää tilastoaineistoa
- Tilastoaineisto pitää uudistaa kun tietokantaan on tehty merkittävästi muunnoksia.
- Tilastoaineistoon kuuluvat esim.
  - taulujen ja indeksien koot
  - sivujen lukumäärä, täyttösuhde
  - sarakkeen pienin ja suurin arvo, mahdollinen arvojen jakautumatieto



Kustannuslaskentaoptimointi

- Esimerkiksi Oraclessa (versio 8) on komento Analyze, jolla kustannuslaskennassa tarvittavan tilastotiedon saa tuotettua
- Analyze table opiskelija estimate statistics;
  - laskee taulukohtaisia tilastotietoja oletuskokoisen otoksen perusteella (otoskoon voi myös antaa)
- Analyse table opiskelija compute statistics for columns aloitusvuosi size 10;
  - laskee sarakkeelle tilastotietoja käymällä läpi koko taulun, tuottaa myös jakautumatiedon (arvoväli jaettu 10 osaan)
- Uudemmissa versioissa gather_X_stats –lauseet, esim gather_table_stats, gather_index_stats,….

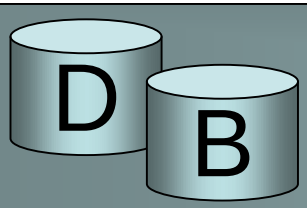


Kustannuslaskentaoptimointi

- Oraclessa optimoijan tekemään suunnitelman saa selville lauseella EXPLAIN PLAN
- esim:

```
EXPLAIN PLAN
SET STATEMENT_ID = 'Raise in Chicago'
INTO plan_table FOR
  UPDATE emp SET sal = sal * 1.10
  WHERE deptno =
    (SELECT deptno FROM dept WHERE loc = 'CHICAGO');
```

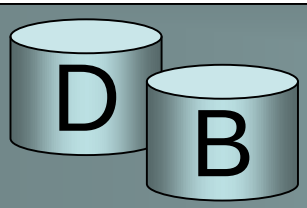
- Yllä suunnitelma kirjoitetaan tässä plan_table nimiseen tauluun. Taulu täytyy olla luotu ennen lauseen käyttöä sen rakenne on seuraava:



Kustannuslaskentaoptimointi

```
CREATE TABLE plan_table (
   statement_id VARCHAR2(30),
   timestamp DATE,
   remarks VARCHAR2(80),
   operation VARCHAR2(30),
   options VARCHAR2(30),
   object_node VARCHAR2(128),
   object_owner VARCHAR2(30),
   object_name VARCHAR2(30),
   object_instance NUMERIC,
   object_type VARCHAR2(30),
   optimizer VARCHAR2(255),
   search_columns NUMERIC,
   id NUMERIC,
   parent_id NUMERIC,
   position NUMERIC,
   cost NUMERIC,
   cardinality NUMERIC,
   bytes NUMERIC,
   other_tag VARCHAR2(255),
   other LONG);
```

tieto taulussa puumaisena rakenteena, parent_id kertoo mikä operaatio suoritetaan rivillä kuvattavan jälkeen



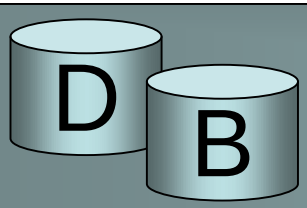
Kustannuslaskentaoptimointi

- esim hierarkkinen kysely

```
SELECT LPAD(' ',2*(LEVEL-1))||operation operation, options, object_name, position
FROM plan_table
START WITH id = 0 AND statement_id = 'Raise in Chicago'
CONNECT BY PRIOR id = parent_id AND statement_id = 'Raise in Chicago';
```

voisi tuottaa tuloksen

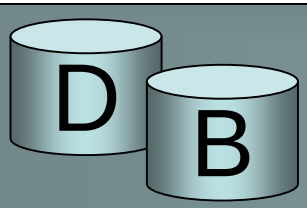
```
OPERATION                      OPTIONS     OBJECT_NAME     POSITION
-------------------------------------------------------------------
UPDATE STATEMENT                                                  1
    FILTER                                                        0
      TABLE ACCESS    FULL        EMP                             1
      TABLE ACCESS    FULL        DEPT                            2
```



Kustannuslaskentaoptimointi

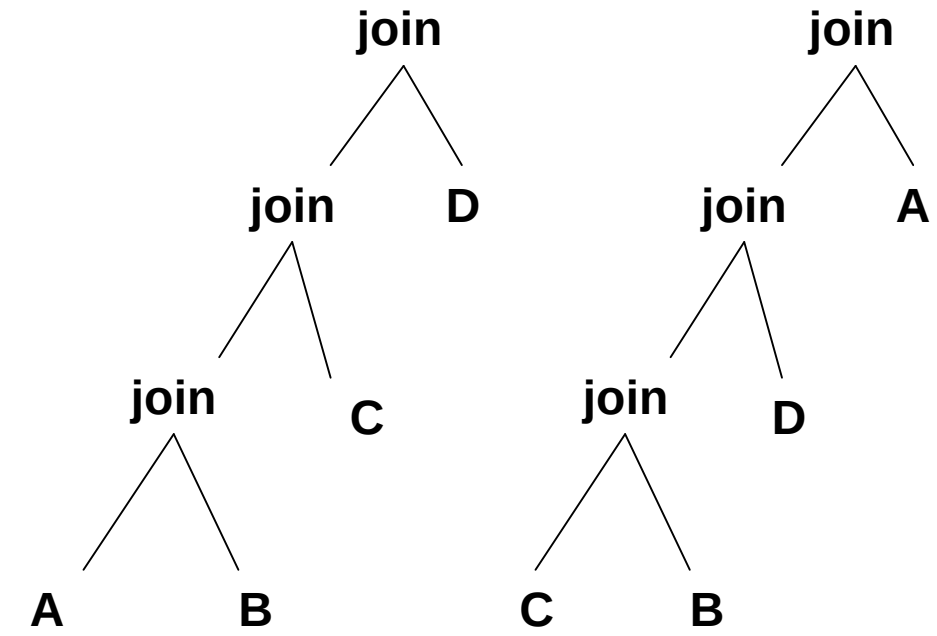
- Kustannuslaskentaoptimoinnissa tuotetaan vaihtoehtoisia toteutussuunnitelmia ja lasketaan niille kustannus.
- Se, jonka kustannus on pienin valitaan.
- Kaikkia vaihtoehtoja ei lasketa,
  - Esimerkiksi useiden liitosten järjestyksen määräämiseksi käytetään ns. vasensyvää puuta (left deep tree). Tässä liitosrakenne on muotoa

    ((A join B) join C) join D ...
  - eli valmiiseen liitostulokseen liitetään taulu kerrallaan
  - lokaalia optimointia: mikä liitos kannattaa tehdä ensin, mikä kannattaa liittää sitten tämän tulokseen, jne
  - liitosrakenteita muotoa (A join B) join (C join D) ei edes arvioida



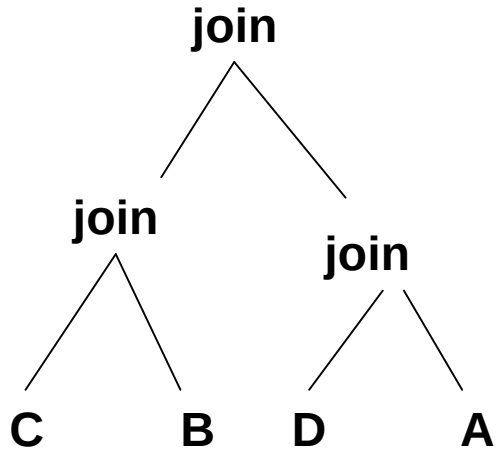
Kustannuslaskentaoptimointi

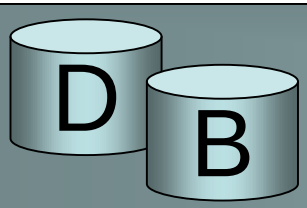
mahdollisia liitosjärjestyksiä tauluille A,B,C,D



vasensyvät puut

ei mahdollinen liitosjärjestys tauluille A,B,C,D





Kustannuslaskentaoptimointi

- Vaikka optimoijalla onkin tiedossa tilastoaineistoa se voi silti päätyä ratkaisuun, joka ei ole välttämättä paras kyseiseen tilanteeseen
- Monissa tkhj:ssä käyttäjä pystyy kyselyyn upotettujen vihjeiden avulla vaikuttamaan optimoijan toimintaan, alla esimerkki Oracle vihjeestä:

```
SELECT /*+ ORDERED USE_NL(customers) */ accounts.balance,
    customers.last_name, customers.first_name
FROM accounts, customers
WHERE accounts.custno = customers.custno;
```

= Liitä taulut from-osassa annetussa järjestyksessä käyttäen sisäkkäisiä silmukoita customers sisempänä tauluna