

Kyselypuut ja ekvivalenssi

Sisäisessä esityksessä kyselyt esitetään kyselypuuna

- lehdissä taulut
- juuressa lopputulos
- välisoimuina 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
 $\sigma_{c_1} \wedge \sigma_{c_2} \wedge \dots \wedge \sigma_{c_k} (R) \equiv \sigma_{c_1}(\sigma_{c_2}(\dots \sigma_{c_k}(R) \dots))$
- Valinnat ovat vaihdannaisia (commutative)
 $\sigma_{c_1}(\sigma_{c_2}(R)) \equiv \sigma_{c_2}(\sigma_{c_1}(R))$
- Peräkkäisten projektioiden lopputuloksen määrä viimeisin projektiio
 $\pi_{a_1}(R) \equiv \pi_{a_1}(\pi_{a_2}(\dots(R)))$

Kyselypuut ja ekvivalenssi

- Ristitulo ja liitos ovat kommutatiivisia, osapuolten järjestys voidaan vaihtaa (kumpi on ulompi)
 - $R \times S \equiv S \times R$
 - $R \bowtie S \equiv S \bowtie R$
- Ristulo ja liitos ovat assosiativiset (suoritusjärjestys on vaihdettavissa)
 - $R \times (S \times T) \equiv (R \times S) \times T$

Kyselypuut ja ekvivalenssi

- Projektion ja valinnan järjestys on vaihdettavissa kunhan kaikki valinnassa esiintyvät sarakkeet ovat mukana projektiossa
 $\sigma_c(\pi_a(R)) \equiv \pi_a(\sigma_c(\dots(R)))$
- Valinta ja ristitulo voidaan yhdistää liitokseksi, jos kyseessä on liitosehto
 $R \bowtie_c S \equiv \sigma_c(R \times S)$
- Valinta voidaan siirtää tapahtuvaksi ennen ristituloa tai liitosta, mikäli se koskee vain näiden argumenttia
 $\sigma_c(R \times S) \equiv \sigma_c(R) \times S$

D B Kyselypuut ja ekvivalenssi

- Yleisesti valinta $\sigma_c(R \times S)$ voidaan jakaa tauluja R ja S koskeviin osiin sekä liitosehtoon eli $\sigma_c(R \times S) \equiv \sigma_{cJ}(\sigma_{cR}(R) \times \sigma_{cS}(S))$
- Projektio voidaan jakaa osiin, jotka suoritetaan ennen ristituloa $\pi_a(R \times S) \equiv \pi_{a1}(R) \times \pi_{a2}(S)$
- Projektio voidaan jakaa osiin, jotka suoritetaan ennen liitosta, jos liitosehdon sarakkeet säilyvät $\pi_a(R \bowtie S) \equiv \pi_{a1}(R) \bowtie \pi_{a2}(S)$

7

D B Heuristisia muokkaussääntöjä

- Konjunkttiiviset valinnat (ja-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

8

D B 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.)
 - 'sarake=vakio' valinta tuottaa tasaisen jakautuman oletuksella rajaussuhteen **1/sarakkeen arvojen lukumäärä**. (jos sarakkeella on hakemisto voi arvojen lukumäärä olla tiedossa, tietotyypin määrittäminen voi rajata arvojoukon esim. numeric(2) enintään 100 arvoa, järjestelmä voi käyttää oletusarvoa esim. 1/10, jos sillä ei ole parempaa tietoa)

9

D B 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.

10

D B Esimerkki kyselypuusta

- Tutkitaan kyselyä

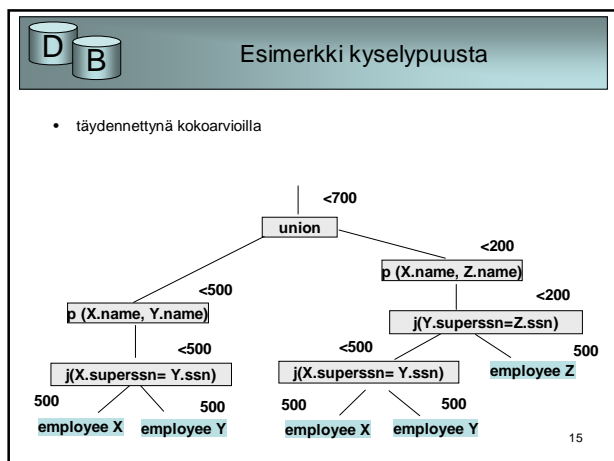
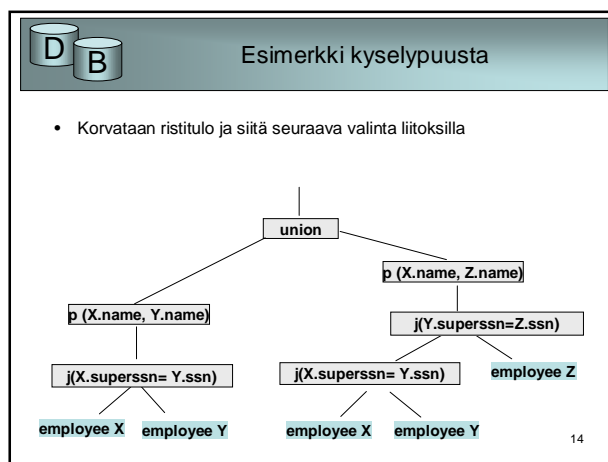
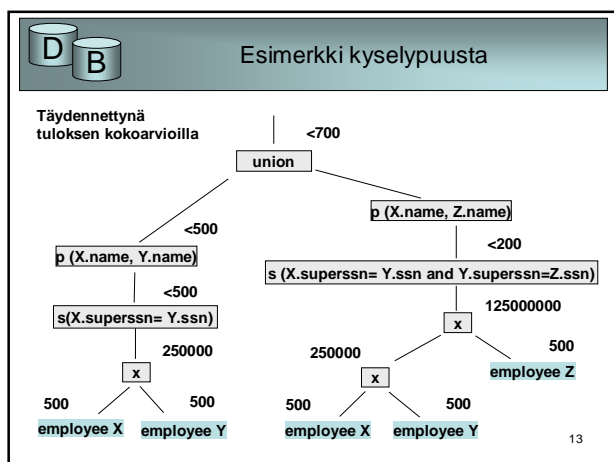
```

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
    
```

11

D B Esimerkki kyselypuusta

12



- 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 jakautumatiieto
- 16

- 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)
 - Analyze table opiskelija compute statistics for columns aloitusvuosi size 10;
 - laskee sarakkeelle tilastotietoja käymällä läpi koko taulun, tuottaa myös jakautumatiiedon (arvoväli jaettu 10 osaan)
 - Uudemmissa versioissa gather_X_stats –proseduurit, esim gather_table_stats, gather_index_stats,....
- 17

- 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 plan_table nimiseen tauluun. Taulu täytyy olla luotu ennen lauseen käyttöä sen rakenne on seuraava:
- 18

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

19

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

20

Kustannuslaskentaoptimointi

- Kustannuslaskentaoptimoinnissa tuotetaan vaihtoehtoisia toteutus suunnitelmia ja lasketaan niille kustannus.
- Se, jonka kustannus on pienin valitaan.
- Kaikkia vaihtoehtoja ei lasketa,
 - Esimerkiksi useiden liitosten järjestyksen määrittämiseksi käytetään ns. **vasensyvä 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
 - liitosrakenneita muotoa **(A join B) join (C join D)** ei edes arvioida

21

Kustannuslaskentaoptimointi

mahdollisia liitosjärjestyksiä tauluille A,B,C,D

```

      join
     /  \
  join   D
 /  \
join  C
/  \
A    B
vasensyvä puut
  
```

```

      join
     /  \
  join   A
 /  \
join  D
/  \
C    B
  
```

ei mahdollinen liitosjärjestys tauluille A,B,C,D

```

      join
     /  \
  join   join
 /  \   /  \
C    B D    A
  
```

22

Kustannuslaskentaoptimointi

- Vaikka optimoijalla onkin tiedossa tilastoaineistoa se voi silti päätyä ratkaisuun, joka ei ole välttämättä paras kyseiseen tilanteeseen
- Monissa tkh: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

23