

Tietokantakyselyjen optimointi relaatiotietokannassa

Olli Rissanen

Kandidaatintutkielma

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

Helsinki, 5. huhtikuuta 2013

Tiedekunta — Fakultet — Faculty		Laitos — Institution — Department	
Matemaattis-luonnontieteellinen		Tietojenkäsittelytieteen laitos	
Tekijä — Författare — Author			
Olli Rissanen			
Työn nimi — Arbetets titel — Title			
Tietokantakyselyjen optimointi relaatiotietokannassa			
Oppiaine — Läroämne — Subject			
Tietojenkäsittelytiede			
Työn laji — Arbetets art — Level		Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Kandidaatintutkielma		5. huhtikuuta 2013	16
Tiivistelmä — Referat — Abstract			
<p>Tutkielmassa tutustutaan tietokantakyselyjen optimointiin relaatiotietokantojen hallintajärjestelmien osalta sekä optimoinnin vaikutukseen kyselyjen suorituskyyvyssä.</p>			
Avainsanat — Nyckelord — Keywords			
Information systems Query optimization			
Säilytyspaikka — Förvaringsställe — Where deposited			
Muita tietoja — Övriga uppgifter — Additional information			

Sisältö

1	Johdanto	3
2	workname: Taustaluku	4
2.1	Relaatiomalli	4
2.2	Optimoijan tavoitteet	5
2.3	Optimoijan historia	6
3	Tietokantakyselyn prosessointi	6
3.1	Kyselyn jäsentäminen	6
3.2	Kyselyn optimointi	8
3.3	Kyselyn suorittaminen	8
4	Optimoijan toiminta	9
4.1	Kyselysuunnitelmien tuottaminen	9
4.2	Kyselysuunnitelmien kustannusarviointi	11
4.3	Tulosjoukon koon arviointi	11
4.4	Systeemitaulustoon tallennettu tilastotieto	12
5	Alikyselyt	13
6	Yhteenveto	14
	Lähteet	14

1 Johdanto

Modernit järjestelmät lisäävät jatkuvasti tietokantojen työtaakkaa tiedon määrän kasvaessa. Jotta tiedosta saadaan mahdollisimman paljon irti, tarvitaan tiedon hallitsemiseen yhä tehokkaampia työkaluja. Tietokannan suorituskyky on tärkeää koko järjestelmän suorituskyvyn osalta, sillä tiedon lukeminen massamuistista on hidasta verrattuna rekistereiden tai välimuistin käyttöön. Optimoimalla tietokantakyselyjen suoritusta voidaan vaikuttaa suoritettujen operaatioiden määrään sekä muistialueen kokoon ja siten vähentää tietokannan vasteaikaa sekä resurssien käyttöä. [MKR12]

Tietokantaa käytetään tietokannan hallintajärjestelmällä, joka on koelma ohjelmia tiedon tallentamiseen, muokkaamiseen, analysointiin ja keräämiseen tietokannasta. Hallintajärjestelmää käytetään kyselykielellä, joista esimerkiksi SQL [CAE⁺76] on suunniteltu relaatiotietokantojen hallintajärjestelmille. Hallintajärjestelmän vastuulla on kyselyn muuttaminen tietokannan ymmärtämään muotoon säilyttäen kyselyn alkuperäisen tarkoituksen. Kyselyn optimointi on toteutettu automaattisena toimenpiteenä hallintojärjestelmän sisältämässä kyselyn optimoijassa, ja kaikista hallintajärjestelmän komponenteista optimoijalla on suurin merkitys tietokannan suorituskykyyn. [MKR12] Kyselyn optimoijan tavoitteena on minimoida itse optimointiin käytetty aika ja maksimoida optimoinnista saatu hyöty. [JK84]

Optimoija toimii rakentaen kyselyä vastaavat kyselysuunnitelmat, arvioimalla niiden suorituskyvyn ja valitsemalla niistä tehokkaimman. [OL90] Kyselysuunnitelma sisältää sarjan algebrallisia operaatioita tietokannan relaatioille jotka tuottavat tulokseksi halutun vastauksen. Tietokantakyselyä vastaavia kyselysuunnitelmia voi olla useita, sillä kyselyjen algebralliset esitykset voidaan usein esittää monena loogisesti vastaavana esityksenä. [JK84] Algebrallista operaatiota kohden voi myös löytyä useita toteutuksia, kuten

join-operaatiota toteuttavat merge join ja hash join. Saman kyselyn tuottamat kyselysuunnitelmat voivat olla suorituskyyvyltään jopa eri suuruusluokassa. [Ioa96, Ora13]

Optimointi on vaikea hakuongelma, jossa hakualue voi nousta erittäin suureksi. Haasteeksi nousee kyselysuunnitelmien luominen ja niiden suorituskyyvyn ennustaminen. Kaikkien mahdollisten kyselysuunnitelmien luominen on usein liian hidasta, joten optimoijan tulee valita pienin mahdollinen hakualue joka pitää sisällään halvimmat suunnitelmat. [Cha98] Suorituskyyvyn ennustamisen ja hakualueen rajauksen lisäksi optimoija tarvitsee tehokkaan algoritmin koko hakualueen läpikäymiseen. On epärealistista odottaa kyselyn optimoijan aina löytävän parhaan kyselysuunnitelman, ja onkin tärkeämpää välttää huonoimpia suunnitelmia ja löytää hyvä suunnitelma. [RG03]

Kappale 2 sisältää esitiedot kyselyn optimoijan toiminnalle, jonka lisäksi kappaleessa perehdytään optimoijan historiaan. Kappale 3 tiivistää tietokantakyselyn prosessoinnin hallintajärjestelmän sisällä ja esittelee optimoijan toiminnan pääpiirteittäin kyselyn prosessoinnissa. Optimoijan toimintaan tarkemmin syvennyttään kappaleessa 4, jossa käsitellään kyselysuunnitelman luomista, kustannusarviointia ja parhaan kyselysuunnitelman valintaa. Kyselyn sisältämien alikyselyjen prosessointi käydään läpi kappaleessa 5.

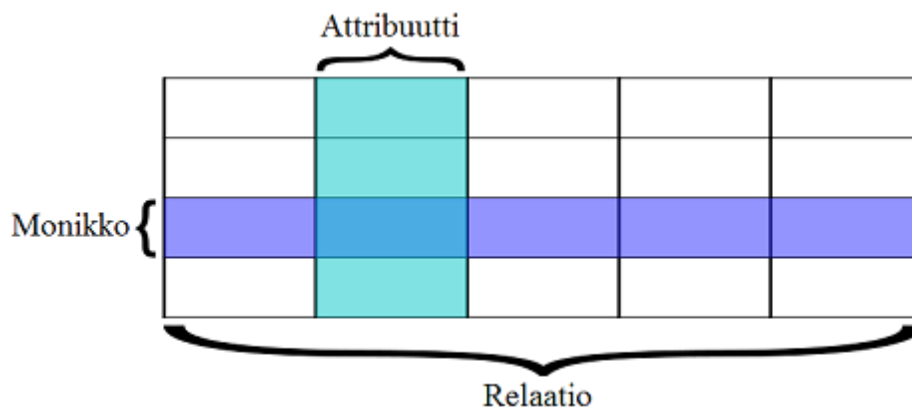
2 workname: Taustaluku

2.1 Relaatiomalli

Relaatiotietokanta on relaatiomalliin [Cod70] perustuva tietokanta. Relaatiomallin keskeinen piirre on kaiken datan esittäminen n-paikkaisen karteesisen tulon osajoukkona, ja se tarjoaa deklarativisen menetelmän datan ja kyselyjen määrittämiseen. Relaatiomalli koostuu attribuuteista, monikoista ja

relaatioista. Matemaattisessa määritelmässä attribuutti on pari joka sisältää attribuutin nimen ja tyyppin sekä jokaiseen attribuuttiin liittyy sen arvojoukko. Monikko on järjestetty joukko attribuuttien arvoja. Relaatio koostuu otsakekeesta ja sisällöstä, jossa otsake on joukko attribuutteja ja keho on joukko monikkoja. Relaation otsake on myös jokaisen monikon otsake. Visuaalisessa esityksissä relaatio on taulukko ja monikko taulukon rivi.

Kuva 1: Relaatiomalliin perustuva tietokanta



2.2 Optimoijan tavoitteet

Tietokantakyselyjen optimoinnilla viitataan tietokantakyselyn suorittamiseen mahdollisimman tehokkaasti. Optimoinnin tavoitteena on joko maksimoida suoritussyky annetuilla resursseilla tai minimoida resurssien käyttö. Jarke ja Koch määrittivät artikkelissaan [JK84] mitattaviksi resursseiksi suorittimen ja muistin käytön sekä kommunikointikustannukset. Muistin käyttö jakautuu tallennuskustannukseen sekä ulkomuistiin pääsyn kustannukseen. Tallennuskustannuksella tarkoitetaan ulkomuistin sekä puskurimuistin käyttöä, ja se tulee aiheelliseksi kun muistin käyttö aiheutuu pullonkaulaksi. Kommunikointikustannukset käsittävät tiedon viennin tallennuspaikasta laskentapaikkaan ja edelleen tulosten esityspaikkaan. Ne jakautuvat kommunikaatioväylän

käyttökustannukseen ja tiedonsiirrosta aiheutuvaan suorittamisen viiveeseen.

Resurssin merkitys riippuu tietokantatyypistä. Jarke ja Koch erittelivät hajautetuissa tietokannoissa kommunikointikustannukset hallitseviksi kustannuksiksi, kun taas paikallisesti hajautetuissa tietokannoissa kaikilla resursseilla on sama painoarvo. Jarken ja Kochin mukaan keskitetyissä tietokannoissa ulkomuistiin pääsyn kustannus ja prosessorin käyttö ovat oleellisia. Tässä tutkielmassa keskitymme erityisesti keskitettyjen tietokantojen optimointiin.

2.3 Optimoijan historia

Kun Coddin relaatiomallin pohjalle luotiin sitä toteuttavia kieliä, nousi vuonna 1974 kehitetty SQL nopeasti käytetyimmäksi. SQL:n myötä optimoinnista tuli osa järjestelmän toiminnallisuutta, sillä SQL tarjosi deklaraatiivisuutensa vuoksi useita tapoja toteuttaa kukin kysely. Ennen relaatiomallin käyttöä optimointi tehtiin käsin. Selinger et al esittelivät artikkelissaan [SAC⁺79] System R-tietokantaohjelmiston OPTIMIZER-komponentin, joka on ensimmäisiä relaatiomalliin pohjautuvia kyselyn optimoijia. System R:n esittelemä dynaaminen ohjelmointi on käytössä eri muodoissa käytännössä kaikissa kaupallisissa järjestelmissä. [Ioa96]

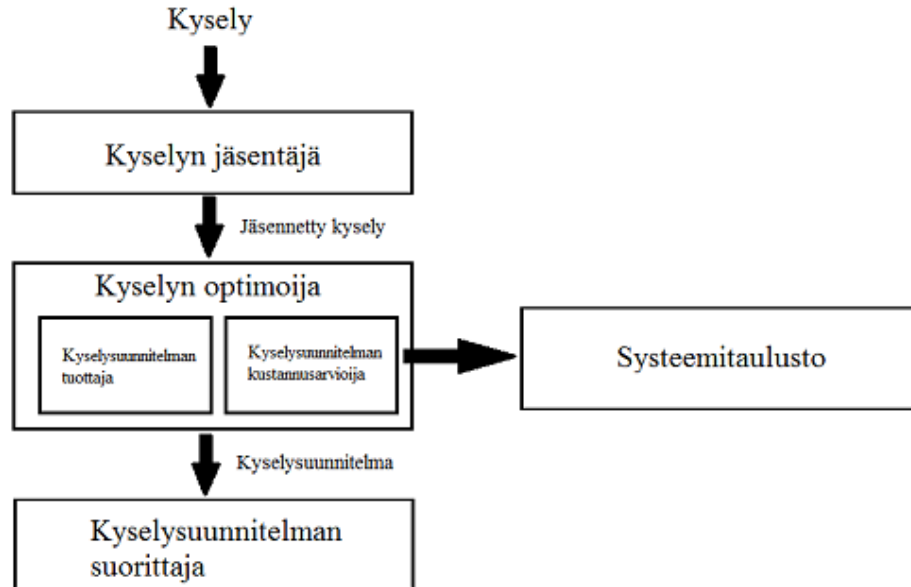
3 Tietokantakyselyn prosessointi

Tietokannan hallintajärjestelmän suorittama SQL-kyselyn prosessointi sisältää useita vaiheita. Kuva 2 esittää kyselyn optimointikerroksen ja suorituserroksen hallintajärjestelmän sisällä.

3.1 Kyselyn jäsentäminen

Kyselyn prosessointi alkaa kyselyn jäsentäjän suorittamalla kyselyn syntaksin ja semantiikan oikeellisuuden validoinnilla. [Ora09] Syntaksin validoinnissa

Kuva 2: Kyselyn jäsentäminen, optimointi ja suoritus



jäsentäjä tarkastaa kyselyn lauseopin oikeellisuuden. Semantiikan validoinnissa tarkastetaan objektien aitous, kyselyn yksiselitteisyys, oikeus haettavaan tietoon ja muuttujien tyyppien sopivuus sarakkeiden tyyppeihin. Useat hallintajärjestelmät myös tallentavat kyselyt validoinnin jälkeen talteen, jotta samaa kyselyä ei tarvitse jäsentää ja optimoida uudelleen. Yksi esimerkiksi kyselyjä tallentavasta hallintajärjestelmästä on Oracle Database, jossa tallennuspaikkaa kutsutaan nimellä Shared Pool. [Ora05]

Seuraavaksi jäsentäjä jakaa kyselyn lohkoihin(block) siten, että yhdessä lohkoissa on täsmälleen yksi SELECT-lause, yksi FROM-lause ja korkeintaan yksi WHERE-, GROUP BY- ja HAVING-lause. [RG03] Kyselyn mahdollisesti sisältämät alikyselyt muodostavat kukin oman lohkonsa. (Alikyselyistä enemmän kappaleessa n?)

Jokainen lohko jäsennetään puuksi, joka on kyselyn algebrallinen esitysmuoto. [MJ12] Puun solmu sisältää yhden operaation kyselyn suorittamiseksi,

ja sillä on nolla tai useampi alisolmaa joiden ulostuloa(output) käytetään sen syötteenä. Esimerkiksi join-operaatiossa solmulla on kaksi alisolmaa, joille toteutetaan join-operaatio ja sort-operaatiolla on yksi alisolmu joka järjestetään. Lehtisolmut ovat solmuja jotka suorittavat hakuja (scan) levyltä ja palauttavat saadut tulokset. Puu suoritetaan lehtisolmuista juureen.

Tämän jälkeen kysely uudelleenkirjoitetaan (rewrite)[Ioa96] valitsemalla hakumetodit (access method), liittämisjärjestyksen (join orders) ja liittämistavat (join methods) tietokannan käyttämien heuristiikkojen pohjalta. Monimutkaisille kyselyille sovelletaan myös muunnossääntöjä. [MJ12] Uudelleenkirjoituksen tarkoitus on helpottaa optimoijan työtä parhaan kyselysuunnitelman valinnassa.

3.2 Kyselyn optimointi

Kun kysely on uudelleenkirjoitettu, lähetetään se kyselyn optimoijalle. Kyselyn optimoija hakee systeemitaulustosta tilastotietoa kyselyyn liittyvistä relaatioista ja relaatioita vastaavista kyselysuunnitelmista. Optimoija määrittää lohkojen suoritusjärjestyksen ja prosessoi jokaisen lohkon FROM-lauseeseen liittyvät relaatiot. Mikäli lohkoissa on useampi relaatio, arvioi optimoija join-operaation permutaatioiden suorituskyyvyt. Optimoija rakentaa lohkoja vastaavat kyselysuunnitelmat käyttäen systeemitauluston tilastotietoa ja valitsee niistä pienimmän kustannuksen sisältävän kyselysuunnitelman. (Miten viitataan koko kappaleeseen?) ([SAC⁺79])

3.3 Kyselyn suorittaminen

Seuraavassa vaiheessa kysely suoritetaan käyttämällä optimoijan tuottamaa kyselysuunnitelmaa. Suorittamisessa kyselysuunnitelma muutetaan suoritettavaksi konekieleksi ja kyselyn lähteen mukaan joko suoritetaan tai

tallennetaan muistiin myöhempää suorittamista varten.

4 Optimoijan toiminta

Kyselyn optimoijan tulee arvioida kustannus jokaiselle kyselysuunnitelmalle. Kustannusarviointi koostuu kahdesta vaiheesta: ensiksi arvioidaan jäsennetyn puun jokaisen alkion operaation suorittamiseen kuluva aika. (pipelining, tempfirelaatiot.) Tämän jälkeen arvioidaan jokaisen alkion tulosjoukon koko, sekä lisäksi tarkastetaan onko tulosjoukko järjestetty. Solmun tulosjoukko on ylisolmun syöte, joten sen koko ja järjestys vaikuttavat suoraan ylisolmun arviointiin.

4.1 Kyselysuunnitelmien tuottaminen

Kyselysuunnitelma koostuu laajennetusta relaatioalgebrapuusta, jossa jokainen solmu kuvaa algebrallista operaatiota. Solmuun on liitetty tieto käytettävästä hakumetodista tiedon hakemiseen taulusta ja suoritustodista relaatio-operaation suoritukseen. Tutkitaan seuraavaa SQL-kyselyä:

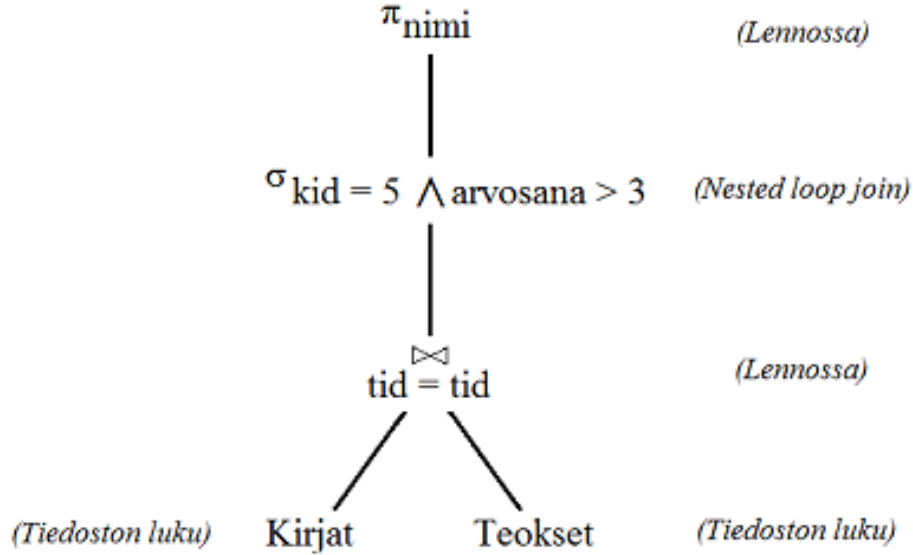
```
SELECT K.nimi
FROM Kirjat K, Teokset T
WHERE K.tid = T.tid AND K.kid = 5 AND T.arvosana > 3
```

Kysely voidaan tulkita relaatioalgebrassa seuraavasti:

$$\pi_{nimi}(\sigma_{kid=5 \wedge arvosana > 3}(Kirjat \bowtie_{tid=tid} Teokset))$$

Kysely voidaan suorittaa esimerkiksi käyttäen join-operaatioissa nested loop join-algoritmia, jonka jälkeen jokaiselle riville suoritetaan valinta ja projektio. Kyselyä vastaava kyselysuunnitelma on siten seuraava:

Kuva 3: Kyselysuunnitelma erimerkkikyselylle



Join-operaatiossa voidaan käyttää hyväksi putkitustekniikkaa (pipelining) [?], jossa operandi putkitetaan suorituksen aikana toiseen ilman väliaikaista taulua tuloksen tallentamiseen. Ilman putkitusta väliaikaiset tulokset tulee materialisoida. Putkitusta käytetään usein peräkkäisten join-operaatioiden toteuttamiseen, jolloin jokainen operaation palauttama operandi voidaan syöttää seuraavan operaation parametriksi. Putkitusta voidaan käyttää operaation suorittavan algoritmin salliessa sen.

Optimoija voi suorittaa usean peräkkäisen join-operaation liittämisen monella eri tavalla. Tutkitaan kyselyä $A \bowtie B \bowtie C \bowtie D$. Left-deep puuksi kutsutaan tapaa liittää kyselyt seuraavasti: $((A \bowtie B) \bowtie C) \bowtie D$, jossa oikea lapsi on aina kantataulu. Optimoijat käyttävät usein pelkästään left-deep puita, sillä eri tapoja liittää join-operaatioita on join-operaatioiden määrän kasvaessa liikaa. Toinen syy on left-deep puiden mahdollistama putkitus. [RG03]

Ionaddis tutki artikkelissaan [IK91] left-deep puiden ja lehtevien (bushy) puiden suorituskykyeroja, ja huomasi lehtevien puiden optimoinnin olevan helpompaa ja sen tuottavan parempia tuloksia.

4.2 Kyselysuunnitelmien kustannusarviointi

4.3 Tulosjoukon koon arviointi

Operaation kustannus riippuu syötteen koosta. Tutkitaan seuraavaa tapausta:

```
SELECT attribuutit
FROM relaatiot
WHERE ehto 1  $\wedge$  ehto 2  $\wedge$  ...  $\wedge$  ehto n
```

Kyselyn palauttamien monikkojen maksimimäärä on relaatioiden karteellinen tulo. Jokainen WHERE-ehto harventaa monikkojen määrää. WHERE-ehdon vaikutusta tulosjoukon kokoon voidaan mallintaa lisäämällä jokaiseen ehtoon vähennyskerroin, joka on oletettu suhde lähtöjoukosta tulosjoukkoon vain kyseisen ehdon osalta. Tulosjoukon koko voidaan siten arvioida kertomalla maksimijoukko vähennyskertoimien tulolla. [RG03]

WHERE-lauseen ehtojen kertoimia voidaan laskea hyödyntämällä systeemitaulustoon tallennettua tilastotietoa. Mikäli relaatiolle ei ole tilastotietoa saatavilla, oletetaan relaation olevan pieni ja käytetään määritettyä vakiota lasketun kertoimen sijaan. [SAC⁺79] Oletetaan seuraavat tiedot:

$NKeys(I)$ = eri avainten lukumäärä indeksissä I

F = vähennyskerroin

sarake = *arvo*

tyyppiselle ehdolle vähennyskerroin voidaan arvioida kaavalla $F = \frac{1}{NKeys(I)}$, jos sarakkeessa on indeksi kyseiselle relaatiolle. [SAC⁺79] Ilman indeksiä kyselyn optimoija käyttää kiinteää arvoa vähennyskerroimen arvioimiseen, joka esimerkiksi System R-relaatiotietokantaohjelmassa on 1/10.

sarake1 = sarake2

tapauksessa voidaan vähennyskerroin arvioida kaavalla $F = \frac{1}{MAX(NKeys(I1), NKeys(I2))}$ jos kummassakin sarakkeessa on indeksi. Lisäksi oletetaan, että jokaisesta pienemmän indeksin arvoa vastaa arvo toisesta indeksistä. Mikäli vain toisessa sarakkeessa on indeksi, voidaan kustannus laskea aiemmalla kaavalla käyttäen indeksin omaavaa saraketta. Mikäli kummassakaan sarakkeessa ei ole indeksiä, arvioidaan arvoksi 1/10.

sarake > arvo

tapauksessa voidaan käyttää kaavaa $F = \frac{\text{suurin avain} - \text{arvo}}{\text{suurin avain} - \text{pienin avain}}$. Mikäli sarake ei ole aritmeettinen tai arvoa ei tiedetä käytetään vakiona arvoa 1/3.

sarake IN (lista arvoja)

$F = koko(lista) \times (sarake = arvo)$

4.4 Systeemitaulustoon tallennettu tilastotieto

Kustannusten arviointi vaatii tilastotietoa. [RG03]

todo: täsmennä ja siisti (sis. systeemitaulustoon tallennetun tilastotiedon käytön) Tilastotieto on.. Tilastotiedoilla lasketaan kyselysuunnitelmien kustannusarvio ja valitaan pienimmän kustannusarvion omaava suunnitelma. Tietoa ei tallenneta jatkuvasti, sillä se aiheuttaisi esimerkiksi rinnakkaisuusongelmia. Tietokannan valvoja triggeriöi päivityksen esim. komennolla optimize

db. Tilastotiedon harvan päivityksen takia valituksi ei aina välttämättä tule tehokkain suunnitelma.

5 Alikyselyt

Kyselyn predikaatin operandin ollessa toinen kysely puhutaan alikyselystä. Alikyselyt suoritetaan..

Alikysely voi palauttaa yhden arvon tai listan arvoja riippuen sen operandista. Tutkitaan seuraavaa kyselyä:

```
SELECT nimi
FROM Kirjat
WHERE hinta =
(SELECT AVG(hinta)
FROM Kirjat)
```

Esimerkkikyselyssä alikysely palauttaa yhden arvon, AVG(hinta). Tämä arvo lisätään ylikyselyyn predikaatiksi niinkuin se olisi ollut osa ylempää kyselyä. Esimerkkitapauksessa keskiarvon ollessa 10 predikaatti on ”hinta = 10”.

Mikäli alikysely palauttaa useamman arvon, kuten seuraavassa tapauksessa:

```
SELECT nimi
FROM Kirjat
WHERE hinta IN (SELECT hinta
FROM Kirjat
WHERE genre = 'scifi')
```

käytetään alikyselyn palauttamaa listaa ylikyselyn predikaattina. Esimerkki-

kyselyn palauttaessa listan (5,15) vastaava ylikyselin predikaatti on ”hinta
IN (5,15).

6 Yhteenveto

Lähteet

- [CAE⁺76] Chamberlin, D.D., M.M. Astrahan, K.P. Eswaran, P. P. Griffiths, R.A. Lorie, J. W. Mehl, P. Reisner ja B.W. Wade: *SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control*. IBM Journal of Research and Development, 20(6):560–575, 1976, ISSN 0018-8646.
- [Cha98] Chaudhuri, Surajit: *An overview of query optimization in relational systems*. Teoksessa *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, sivut 34–43. ACM, 1998.
- [Cod70] Codd, Edgar F: *A relational model of data for large shared data banks*. Communications of the ACM, 13(6):377–387, 1970.
- [IK91] Ioannidis, Yannis E ja Younkyung Cha Kang: *Left-deep vs. bushy trees: An analysis of strategy spaces and its implications for query optimization*. Teoksessa *ACM SIGMOD Record*, nide 20, sivut 168–177. ACM, 1991.
- [Ioa96] Ioannidis, Yannis E: *Query optimization*. ACM Computing Surveys (CSUR), 28(1):121–123, 1996.
- [JK84] Jarke, Matthias ja Jurgen Koch: *Query optimization in database systems*. ACM Computing surveys (CsUR), 16(2):111–152, 1984.

- [MJ12] Mahajan, S.M. ja V.P. Jadhav: *An analysis of execution plans in query optimization*. Teoksessa *Communication, Information Computing Technology (ICCICT), 2012 International Conference on*, sivut 1–5, 2012.
- [MKR12] Mor, Jyoti, Indu Kashyap ja RK Rathy: *Analysis of Query Optimization Techniques in Databases*. *International Journal*, 47, 2012.
- [OL90] Ono, Kiyoshi ja Guy M Lohman: *Measuring the complexity of join enumeration in query optimization*. Teoksessa *Proceedings of the 16th International Conference on Very Large Data Bases*, sivut 314–325. Morgan Kaufmann Publishers Inc., 1990.
- [Ora05] Oracle: *Understanding Shared Pool Memory Structures*, syyskuu 2005. <http://www.oracle.com/technetwork/database/focus-areas/manageability/ps-s003-274003-106-1-fin-v2-128827.pdf>.
- [Ora09] Oracle: *Oracle Database Online Documentation 11g Release 1*, 2009. http://docs.oracle.com/cd/B28359_01/server.111/b28318/sqlplsql.htm.
- [Ora13] Oracle: *MySQL 5.0 Reference Manual*, 2013. http://docs.oracle.com/cd/E17952_01/refman-5.0-en/controlling-optimizer.html.
- [RG03] Ramakrishnan, R. ja J. Gehrke: *Database Management Systems*. McGraw-Hill international editions: Computer science series. McGraw-Hill Education, 2003, ISBN 9780072465631. <http://books.google.fi/books?id=JSVhe-WLGZ0C>.

- [SAC⁺79] Selinger, P Griffiths, Morton M Astrahan, Donald D Chamberlin, Raymond A Lorie ja Thomas G Price: *Access path selection in a relational database management system*. Teoksessa *Proceedings of the 1979 ACM SIGMOD international conference on Management of data*, sivut 23–34. ACM, 1979.