

Tietokantakyselyjen optimointi relaatiotietokannassa

Olli Rissanen

Kandidaatintutkielma

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

Helsinki, 24. maaliskuuta 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		24. maaliskuuta 2013	11
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 toiminta	6
3	Kyselyn optimointi	7
3.1	Kyselysuunnitelmien tuottaminen	8
3.2	Kyselysuunnitelmien kustannusarviointi	9
3.3	Tulosjoukon koon arviointi	9
3.4	Systeemitaulustoon tallennettu tilastotieto	10
4	Optimoijan käyttäminen (hints)	10
5	Käyttöesimerkki, suorituskykyvertailu	10
6	yhteenveto	10
	Lähteet	10

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 on suunniteltu relaatiotietokantojen hallintajärjestelmille. Hallintajärjestelmän vastuulla on kyselyn muuttaminen tietokannan ymmärtämään muotoon säilyttäen kyselyn alkuperäisen tarkoituksen. Kyselyn optimointi on toteutettu automaattisena toimenpiteenä tietokannan 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 etsien kyselyä vastaavat kyselysuunnitelmat ja valitsemalla niistä tehokkaimman. 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 suorituskyvyltään jopa eri suuruusluokassa. [Ora13]

Optimointi on vaikea hakuongelma, jossa hakualue voi nousta erittäin suureksi. [Cha98] Haasteeksi nousee kyselysuunnitelmien luominen ja niiden suorituskyvyn ennustaminen. Kaikkien mahdollisten kyselysuunnitelmien luominen on usein liian hidasta, joten optimoijan tulee valita pienin mahdollinen hakualue joka pitää sisällään halvimmat suunnitelmat. Suorituskyvyn 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 rakenteeseen hallintajärjestelmän sisällä. Kappale 3 käsittelee optimointiprosessin vaiheita yksityiskohtaisesti. Kappaleissa 4 ja 5 esitellään kyselyn optimoijan käyttöä MySQL-hallintajärjestelmällä.

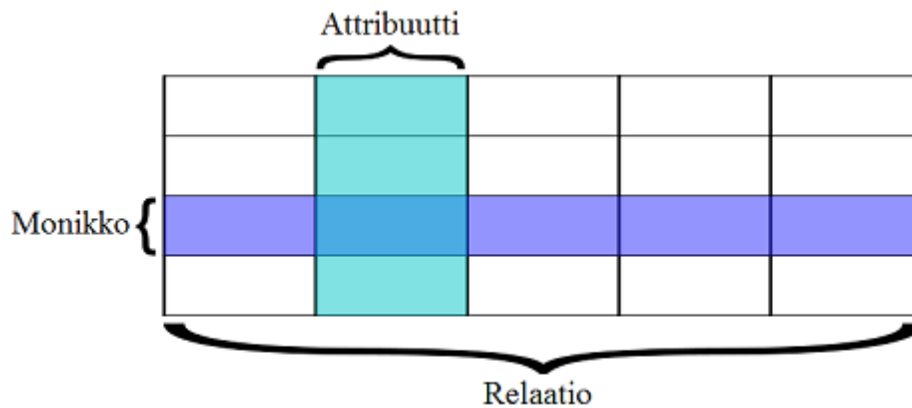
2 workname: Taustaluku

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. Relatio koostuu otsakeesta 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 suorituskky annetuilla resursseilla tai minimoida resurssien käyttö. Mitattavia resursseja ovat suorittimen ja muistin käyttö sekä kommunikointikustannukset. [JK84] 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. Hajautetuissa tietokannoissa hitailla yhteysväylillä kommunikointikustannukset hallitsevat kustannuksia. Paikallisesti hajautetuissa tietokannoissa kaikilla resursseilla on sama

painoarvo. Keskitetyissä tietokannoissa ulkomuistiin pääsyn kustannus ja prosessorin käyttö ovat oleellisia. [JK84] Tässä tutkielmassa keskitymme erityisesti keskitettyjen tietokantojen optimointiin.

2.3 Optimioijan toiminta

Tietokannan hallintajärjestelmän suorittama SQL-kyselyn prosessointi sisältää useita vaiheita. Kuva 2 esittää kyselyn optimioijan rakenteen hallintajärjestelmän sisällä. Kyselyn prosessointi alkaa kyselyn jäsentäjän suorittamalla kyselyn syntaksin ja semantiikan oikeellisuuden validoinnilla. [Ora09] Syntaksin validoinnissa jäsentäjä tarkastaa kyselyn lauseopin oikeellisuuden. Semantiikan validoinnissa tarkastetaan objektien aitous, kyselyn yksiselitteisyys, oikeus haettavaan tietoon ja muuttujien tyyppin sopivuus sarakkeiden tyyppeihin. Useat hallintajärjestelmät myös tallentavat kyselyt validoinnin jälkeen talteen, jotta samaa kyselyä ei tarvitse jäsentää ja optimoida uudelleen. Yksi esimerkki kyselyjä tallentavasta hallintajärjestelmästä on Oracle Database, jossa tallennuspaikkaa kutsutaan nimellä Shared Pool. [Ora05]

Seuraavaksi jäsentäjä jakaa kyselyn lohkoihin(block) siten, että yhdessä lohossa 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 alisolmu joiden ulostuloa(output) käytetään sen syötteenä. Esimerkiksi join-operaatiossa solmulla on kaksi alisolmuja, 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.

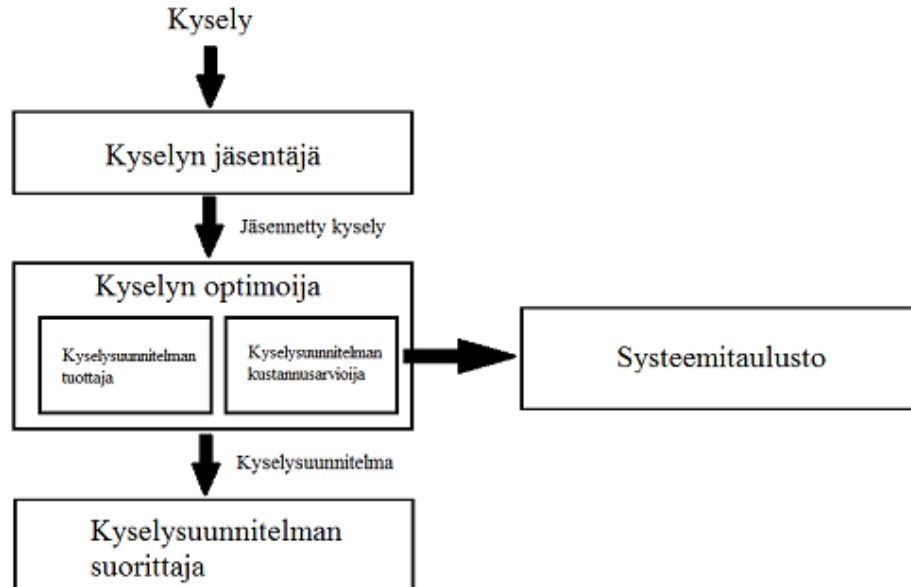
Tämän jälkeen kysely uudelleenkirjoitetaan (rewrite) valitsemalla hakumetodit (access method), liittämisjärjestyksen (join orders) ja liittämistavat (join methods) tietokannan käyttämien heuristiikkojen pohjalta. Monimutkaisille kyselyille sovelletaan myös muunnossääntöjä. [MJ12] Uudelleenkirjoituksen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

Kun kysely on uudelleenkirjoitettu, lähetetään se kyselyn optimoijalle. Kyselyn optimoijan sisääntulo(input) sisältää kyselyn, tietokannan mallin sisältäen taulujen ja indeksien määritelmät sekä tietokannan tilastoja. Kyselystä haetaan predikaatit, indeksit ja liitokset (joins) kyselysuunnitelmaa varten. Tämän jälkeen luodaan kyselyä vastaavat kyselysuunnitelmat ja arvioidaan niistä paras. Optimoija arvioi operaatioiden kustannukset käyttämällä systeemitaulustoon tallennettua tilastotietoa. Seuraavassa vaiheessa kysely suoritetaan käyttämällä optimoijan tuottamaa kyselysuunnitelmaa.

3 Kyselyn optimointi

Kyselyn optimoijan tulee arvioida kustannus jokaiselle kyselysuunnitelmalle. Kustannusarviointi koostuu kahdesta vaiheesta: ensiksi arvioidaan jäsennetyn puun jokaisen alkion operaation suorittamiseen kuluva aika. (pipelining, temppirelaatiot.) 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.

Kuva 2: Kyselyn jäsentäminen, optimointi ja suoritus



3.1 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-operaatiossa nested loop join-algoritmia, jonka jälkeen jokaiselle riville suoritetaan valinta ja projektio.

Kyselyä vastaava kyselysuunnitelma on siten seuraava:

3.2 Kyselysuunnitelmien kustannusarviointi

3.3 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 karteesinen 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.

$$column = value \tag{1}$$

-tyyppiselle ehdolle vähennyskerroin voidaan arvioida kaavalla

$$\frac{1}{NKeys(I)} \tag{2}$$

jos sarakkeessa on indeksi I kyseiselle relaatiolle. Ilman indeksiä kyselyn optimoija käyttää kiinteää arvoa vähennyskertoimen arvioimiseen, kuten 1/10.

3.4 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.

4 Optimoijan käyttäminen (hints)

5 Käyttöesimerkki, suorituskykyvertailu

6 yhteenveto

Lähteet

- [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.
- [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.
- [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/sqlplspl.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-WLGZOC>.