Faktataulun käsittelyn perusasiat

Tapio Lahdenmäki,
IBM

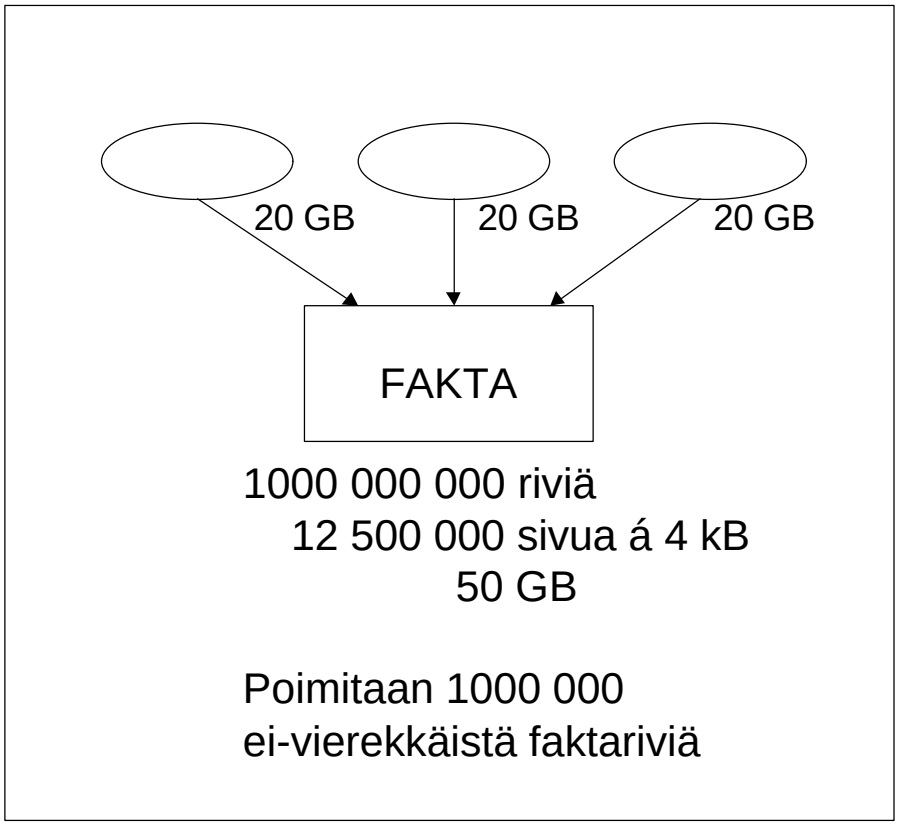
Valtaosa tähtiliitoksen ajasta kuluu tavallisesti faktataulun käsittelyyn. Pitkästä käsittelyajasta voidaan syyttää joko väärän saantipolun valinnutta optimoijaa tai levymuistin fyysisiä ominaisuuksia; ison faktataulun käsittely voi olla tuskallisen hidasta älykkäimmälläkin saantipolulla.

Ralph Kimball on todennäköisesti oikeassa väittäessään, että faktataulun luvun tulee olla tähtiliitoksen viimeinen vaihe: sitä ennen on luettavien faktarivien määrä minimoitava dimensiotaulujen avulla. Relaatiokantajärjestelmän optimoija ei tätä välttämättä aina ymmärrä: se voi valita liitoksen taulujärjestykseksi DIM1, DIM2, FAKTA, DIM3. Optimoijan olettamilla läpäisykertoimilla tämä voi olla nopein vaihtoehto, mutta todellisuudessa DIM1, DIM2, DIM3, FAKTA voi olla paljon nopeampi.

Vaikka taulujärjestys olisi paras mahdollinen, optimoijan valitsema saantipolku voi olla kaukana optimaalisesta. Optimoija ei esimerkiksi osaa aina asettaa taulun läpilukua ja indeksipohjaista hajalukua oikeaan paremmuusjärjestykseen.

Nykyään ei onneksi ole kovin vaikeaa ohjata optimoijaa oikeaan vaihtoehtoon - kunhan joku ensin huomaa optimoijan kömmähdyksen ja näkee oikean saantipolun. Tilanne on pulmallisempi, jos paraskaan käytettävistä olevista saantipoluista ei ole kyllin hyvä.

Se, joka osaa ennustaa taulun peruskäsittelytapojen keston, tulee paremmin toimeen isojen faktataulujen, epätäydellisten optimoijien ja hitaiden levymuistien kanssa.



Kuva: Esimerkki 1

Taulun läpiluku

Levytaltion pyörimisnopeus on valon nopeuden kaltainen raja-arvo. Jos on luettava miljoona uraa (noin 50 gigatavua), minimikesto ilman rinnakkaisajoa on miljoona pyörähdystä eli noin 1 000 000 x 10 ms = 10 000 s. Turvallinen peukalosääntö useimmille levytyypeille on **500 s per gigatavu**. Tähän mahtuu kohtalaisen paljon jonotusta ja muita viiveitä. Ilman rinnakkaistusta 50 gigatavun faktataulun läpiluku voi täten viedä 50 GB x 500 s/GB = 25 000 s, siis seitsemän tuntia.

SQL-lauseen rinnakkaistamista tarkastellaan tuonnempana. Äskettäin julkistetuilla sarjallistetuilla levylaitteilla (IBM Enterprise Storage Server) peräkkäisluku rinnakkaistuu fyysisellä tasolla: Taulun peräkkäiset urat jaetaan eri taltioille ja niitä siirretään keskusmuistiin neljää rinnakkaista väylää pitkin. Tällöin taulun läpilukuaika voi olla **100 s per gigatavu.**

Taulun osan läpiluku

Faktataulun rivit talletetaan useimmiten aikajärjestykseen. Pvm-alkuisen indeksin kautta voidaan tällöin lukea tehokkaasti tietyn pvm-välin faktarivit. Jos indeksin koko on 20 gigatavua ja taulun 50 gigatavua, yksi 50 kuukaudesta saadaan luetuksi peukalosäännön mukaan 700 sekunnissa (1.4 GB x 500 s/GB). Vielä hieman parempaan tulokseen päästään, jos faktataulu on jaettu kuukausikohtaisiin osioihin. Tällöin luetaan läpi yksi osio: 1 GB x 500 s/GB = 500 s.

Hajaluku

Oletetaan, että 50 gigatavun esimerkkitaulumme miljardista rivistä luetaan miljoona ei-vierekkäistä riviä, siis yksi promille. Indeksin käyttö tun-

tuu järkevältä, koska sillä vältetään 999 miljoonan rivin lukeminen.

Indeksistä luetaan vain yhden promillen kokoinen viipale peräkkäiskäsittelyllä. Taulun käsittelyaika riippuu siitä, ovatko luettavat rivit täysin hajallaan. Oletetaan ensin pahin vaihtoehto: täysin satunnainen käsittelyjärjestys.

Hajakäsittelyn fyysisiä raja-arvoja ovat levytaltion pyörimisnopeus ja hakuvarren liike. Jos on luettava miljoona riviä isosta taulusta satunnaisessa järjestyksessä, aikaa kuluu nopeimmilla nykylevyillä (10 000 rpm) 1 000 000 x 10 ms = 10 000 s = 3 h. Turvallinen ja paljon käytetty peukalosääntö on 20 ms per rivi. Tämä kohtalaisen jonotuksen ja rauhallisemman pyörimisnopeuden salliva kaava antaa I/O-ajan arvioksi kuusi tuntia. Taulun arvioitu läpilukuaika oli samaa luokkaa.

Levypolkukuormituksen kannalta hajakäsittely on tässä tapauksessa edullisempaa kuin läpiluku, koska siirrettävän tiedon määrä on pienempi: miljoona sivua eli neljä gigatavua, jos sivun koko on neljä kilotavua. Läpilukuvaihtoehdossa joudutaan siirtämään 50 gigatavua.

Kuusi tuntia on pitkä aika. Onneksi monet optimoijat osaavat nykyään lyhentää hajakäsittelyn kestoa seuraavassa kuvatulla tavalla.

Harppova peräkkäisluku

Usein on järkevää kerätä indeksistä hakuehtoon liittyvät osoittimet ja **lajitella** ne osoitteen mukaan. Vasta sitten luetaan faktataulun rivit. Tällainen harppova peräkkäisluku (skip sequential, list prefetch) lyhentää I/O-aikaa per rivi, koska hakuvarren keskimääräinen liike lyhenee. Joskus keskimääräinen pyöräh-dysviivekin on vähemmän kuin puoli kierrosta Säästö riippuu tietenkin siitä, miten suuri osa sivuista luetaan. Jos luetaan joka toinen sivu, I/O-aika lyhenee 20 millisekunnista per rivi pariin millisekuntiin per sivu.

Esimerkissä 1 luetaan noin joka kymmenes sivu eli vähän yli yksi sivu per ura. Koska hakuvarren keskimääräinen liike putoaa alle millisekunnin ja pyörähdysviivekin hieman lyhenee, I/O-aika per rivi on todennäköisesti viiden ja kymmenen sekunnin välillä. Koko-naiskesto on pari tuntia. Tämä on paras rinnakkaistamaton saantipolku esimerkissä 1.

Rinnakkaistaminen

Läpiluku

Taulun läpilukua voidaan nopeuttaa tuntuvasti SQL-käsittelyn rinnakkaistuksella. Jos 50 gigatavun kanta on kymmenellä taltiolla kymmenen siirtopolun takana, tehtävä kannattaa jakaa kymmeneksi yhtä suureksi osatehtäväksi, jotka voivat edetä toisiaan häiritsemättä. Koko läpiluvun kesto on tällöin vain 1000 s. Näin yksinkertaisessa tilanteessa tietokantajärjestelmän optimoija osaa varmaan pilkkoa yhden SELECT-lauseen kymmeneksi pienemmäksi, joista kullakin on tutkittavan alueen rajaava WHERE-ehto.

SQL-käsittelyn automaattinen rinnakkaistus on tyylikäs asia, mutta siihen liittyy kolme sudenkuoppaa: levyjonotus, suoritinjonotus sekä resurssien jako.

Rinnakkaistamisen rajat

Taltioita ja levypolkuja on oltava riittävästi, jotta alitehtävät voivat todella edetä rinnakkain. Jos koko tietokanta on yhdellä isolla taltiolla, ei levy-I/O:n rinnakkaistaminen onnistu ollenkaan

Yhdestä sivuvirrasta nopeimmat nykysuorittimet selviävät yleensä kunnialla. Peukalosääntömme (500 s per gigatavu) mukaan peräkkäiskäsittelynopeus on 2 MB/s. Millisekunnissa tulee käsitellä 2 000 tavua. Faktarivit ovat usein lyhyitä, joten 2000 tavuun voi mahtua esimerkiksi 50 riviä. Rivin keskimääräinen käsittelyaika saa tällöin olla korkeintaan 20 mikrosekuntia (1 ms / 50 riviä), jotta CPU-aika limittyisi levyajan kanssa, kuten läpiluvun kohdalla yleensä oletetaan. Kymmenestä sivuvirrasta keskusmuistiin syöksyy 20 000 tavua millisekunnissa. Nykysuorittimet eivät oikein ehdi edes hylätä riviä kahdessa mikrosekunnissa. Tarvitaan useita suorittimia, ehkä jopa yksi per sivuvirta.

Jos taltioita ja levypolkuja on juuri ja juuri riittävästi yhden kyselyn tehokkaaseen rinnakkaistamiseen - kuten esimerkissämme - miten muiden samanaikaisten käyttäjien käy? Joutuvatko he odottamaan 15 minuuttia, kunnes esimerkkikysely on suoritettu? Tuskin - ellei käyttöjärjestelmä ole aivan alkeellinen tai prioriteettiasetukset aivan eriarvoistavia. Monen käyttäjän ympäristössä rinnakkaiset voimakkaasti rinnakkaistetut kyselyt kuitenkin häiritsevät toisiaan tavalla tai toisella.

Voiko muuta kuin läpilukua rinnakkaistaa?

Hajakäsittelyn rinnakkaistaminen on lähes yhtä helppoa kuin läpiluvun, jos työ voidaan jakaa osiin indeksiavaimen perusteella. Moni optimoija osaa tämän tehdä, mutta osien koko voi olla vaihtelevampi kuin läpilukutapauksessa.

Läpilukuun verrattuna taltiojonotuksen vaara on suurempi, koska taltioviittaukset ovat satunnaisia. Jos rinnakkaisuusaste on kymmenen, taulun tulee jakautua ainakin 20 taltiolle, jotta jonotusajat pysyisivät siedettävinä. Toisaalta levypolkukuormitus on huomattavasti pienempi kuin peräkkäiskäsittelyssä. Neljä polkua riittäisi varmaan hyvin esimerkkitapauksessa.

Harppova peräkkäisluku ei sulje pois rinnakkaistamista, päinvastoin, se helpottaa työn jakamista yhtä suuriin osiin. Kaikki optimoijat eivät kuitenkaan ole vielä oppineet tätä yhdistelmää, joka esimerkissä 1 tuntuu parhaalta. Jos rinnakkaisuusaste olisi kymme-

nen, miljoonan faktarivin löytämiseen kuluisi vähän yli 2 h / 10 eli suunnilleen varttitunti.

Entä jos varttitunti on kohtuuton kesto tai kymmenestä rinnakkaistehtävästä aiheutuva systeemikuormitus liian korkea?

Puolijohdemuistia levyohjaimeen

Uusimmissa levylaitteissa on paljon puolijohdemuistia, jopa useita gigatavuja. Kun sivu löytyy ohjaimen muistista, levy-I/O:n kesto on vain pari millisekuntia. Olisiko esimerkiksi neljän gigatavun ohjainmuisti se lopullinen ratkaisu, jolla miljoonan eivierekkäisen rivin lukuaika saataisiin siedettäväksi?

Tuskin. Neljän gigatavun puskurimuistilla on dramaattinen vaikutus, jos koko tietokanta on samaa suuruusluokkaa tai jos taulujen aktiivisuus vaihtelee kuten useimmissa operatiivisissa järjestelmissä. Tilanne on huonompi, kun 50 gigatavun taulua luetaan täysin satunnaisesti. Todennäköisyys sille, että tietty sivu on valmiiksi muistissa, on alle 10 prosenttia.

Optimoija ei ole kaikkitietävä

Optimoija voi tähtiliitosta pähkäillessään erehtyä ainakin kolmella tasolla: taulujärjestys, faktataulun saantipolku ja rinnakkaisuusaste. Näistä keskimmäinen on ehkä tavallisin.

Esimerkissä 1 ihminen oivaltaa nopeasti, että harppova peräkkäiskäsittely on paras saantipolku; sen ennuste on pari tuntia ilman rinnakkaistusta. Optimoija saattaisi kuitenkin valita taulun läpiluvun, vaikka sen ennuste on kuusi tuntia ilman rinnakkaistusta. Miksi?

Meidän laskelmamme perustuivat siihen, että WHERE-ehdon läpäisykerroin on yksi promille. Optimoija ei tätä tiedä, emmekä me voi sitä optimoijalle SQL-lauseessa kertoa. Optimoija tietää, että sarakkeessa TUOTE_ID on 100 000 erilaista arvoa. Lisäksi se todennäköisesti tietää yleisimpiin tuotenumeroihin liittyvät rivimäärät sekä jotakin tuotenumeroiden jakautumasta (2 % alle 1 000, 3 % alle 2 000 jne). Kun kysely kohdistuu tuotenumeroon, joka ei ole yleisimpien joukossa, optimoijan estimaatti läpäisykertoimelle voi olla kaukana totuudesta. Esimerkissä 1 liian suuri estimaatti johtaa helposti taulun läpilukuun, koska se voi olla optimoijan laskelmien mukaan nopein vaihtoehto.

Väärän valinnan todennäköisyys kasvaa, kun käytetään arvoväliehtoa (esim. MK > 100000). Sarakearvon jakautumahistogrammi on usein liian karkea.

Operatiivisten relaatiokantojen hoitajat ovat kauan sitten oppineet tulemaan toimeen epätäydellisten optimoijien kanssa. Toistuvien staattisten SQL-kutsujen saantipolkujen viritys on kuitenkin helppoa, kun sitä vertaa tietovarastoille ominaisiin dynaamisiin ja ennalta arvaamattomiin kutsuihin.

Isojen faktatulujen hoitaja tarvitsee seurantavälineen, joka raportoi pitkät kyselyt ja optimoijan niihin valitseman saantipolun. Näiden tietojen perusteella voidaan parantaa tietokantaa ja joissakin tapauksissa myös auttaa optimoijaa. Optimoijan auttaminen on vaikeaa, jos kysely generoi SQL-lauseen; tietovaraston hyväksikäyttäjä jaksaa tuskin kiinnostua riittävässä määrin optimoijan sudenkuopista. Tällöin ainoaksi vaikutusmahdollisuudeksi jää tietokannan parantaminen joko indeksillä tai summataululla. Todella hyvää saantipolkua on vaikea vastustaa.

Johtopäätöksiä

Nykylaitteistoilla ei ilman rinnakkaistamista esimerkissä 1 päästä alle parin tunnin kestoon, vaikka optimoija olisi äärettömän älykäs. Tuloksekas rinnakkaistaminen vaatii useita suorittimia, taltioita ja levypolkuja.

Jos tämä suoritusteho ei riitä, on joko eliminoitava taulurivien luku paksuilla indekseillä tai vähennettävä luettavien rivien määrää summatauluilla. Näitä usein välttämättömiä virityksiä pohditaan Ari Hovin artikkelissa.

Muita ongelmia

Edellä on käsitelty vain tähtiliitoksen viimeistä vaihetta, faktarivien poimintaa. Tätä ennen on jotenkin muunnettava WHERE-ehto joko avainkombinaatioiksi tai osoittimiksi. Marja Kärmeniemi käsittelee artikkeleissaan näihin vaiheisiin liittyviä ongelmia ja nykytuotteiden tarjoamia ratkaisuvaihtoehtoja.

tapio_lahdenmaki@fi.ibm.com