Paksut indeksit ja summataulut

Ari Hovi,
Ari Hovi Oy

Esittelen ensin moniosaiset ja paksut indeksit ja summataulut käsitteinä. Artikkelin loppuosassa käyn läpi miten paksut indeksit ja summataulut auttavat esimerkkikyselymme suorituksessa.

Paksuilla indekseillä tarkoitetaan moniosaisia indeksejä eli indeksejä, jotka koostuvat monesta sarakkeesta ja jotka sisältävät myös haettavaa tietoa, eikä siis vain hakuavaimia. Joskus käytetään myös termiä kattava indeksi (covering index). Kaikki yleiset tietokantatuotteet mahdollistavat paksut indeksit.

Riittääkö tavallinen indeksi

Tavallinen indeksi tarkoittaa tässä yhteydessä indeksiä, joka sisältää WHERE-lauseessa olevia ehtosarakkeita. Se ei siis sisällä haettavaa tietoa, kuten paksu indeksi. Tarkastellaan ensin, miten tavallisella indeksillä tullaan toimeen.

Esimerkissämme on Myynti-taululle määritelty indeksi Aika_ID, Tuote_ID, Liike_ID. Tarkastallaan ensin tällaista moniosaista, vain avaintietoa sisältävää indeksiä eri hakutilanteissa.

Katsotaanpa seuraavaa SQL-käskyä:

```
SELECT myynti_mk, myynti_kpl
FROM myynti
WHERE aika_ID = 23
AND tuote_ID = 3400
AND liike_ID = 7211
```

Tällainen kysely toimii hyvin tehokkaasti, sillä hakuehdot löytyvät kaikki indeksistä. Tarvitaan yksi hajaluku sekä tauluun että indeksiin (taulu on aika_ID-järjestyksessä). Tästä tulee 2 * 20 ms (ks. T. Lahdenmäen artikkeli kohta hajaluku). Sen jälkeen luetaan 1 promille sekä indeksistä että 1 promille taulusta peräkkäislukuna. Luetaan siis 25 MB indeksiä (indeksin koko on 2,5 GB) sekä 60 MB taulua. Kaavalla 0,5 s per megatavu saadaan 40 sekuntia. Tilanne olisi aivan toinen, jos taulurivin luku olisi harppovaa peräkkäislukua tai hajalukua. Jälkimmäisessä tapauksessa I/O-aika olisi 1 000 000 * 20 ms eli 20 000 s, mikä tekee noin 6 tuntia. Tarvittaneen paksua indeksiä taulun luvun välttämiseksi.

Seuraava esimerkki edustaa indeksin alkuosalla hakua:

```
SELECT myynti_mk, myynti_kpl
FROM  myynti
WHERE aika_ID = 23
```

Tämäkin indeksi toimii melko hyvin, sillä indeksin alkuosalla päästään suoraan oikeaan kohtaan ja sitten suorahaut tauluun. Jos tulosjoukko on suuri, voi taulun kohdistuvien hajalukujen I/O-aika kuitenkin kasvaa.

Allaolevassa esimerkissä haetaan indeksin ensimmäisellä ja kolmannella sarakkeella. Tässä haku ei ole niin tehokas, koska indeksistä on luettava peräkkäin kaikki aika_ID:n 23 omaavat ja sitten hylättävä ne, joiden liike_ID ei ole 7211. Tätä kutsutaan englannin kielellä termillä “Index screening”.

```
SELECT myynti_mk, myynti_kpl
FROM myynti
WHERE aika_ID = 23
AND liike_ID = 7211
```

Edellä olevassa esimerkissä luetaan 25 MB indeksiä ajassa 12,5 s ja taulusta 1 000 riviä (tietty päivä eli 1 promille ja tietty liike eli 1 promille siitäkin). Aikaa kuluu 1000 + 2 ms. eli pahimmillaan noin 20 s taululukua.

Moniosaisesta indeksistä ei ole hyötyä, jos haetaan indeksin loppuosan sarakkeilla. Asia on sama, kuin jos yritettäisiin hakea puhelinluettelosta etunimellä. Katsotaan esimerkkiä :

```
SELECT myynti_mk, myynti_kpl
FROM myynti
WHERE liike_ID = 7211
```

Tuotteesta riippuen tapahtuu joko kaikkien indeksisivujen läpiluku (non-matching index scan) liike-ID:n 7211 löytämiseksi tai peräti pahin vaihtoehto eli koko Myynti-taulun läpiluku. Jälkimmäisessä vaihtoehdossa ei siis indeksi tule käyttöön lainkaan. Taulun läpiluku kestäisi kahdeksan tuntia!

Sitten tarkastelemme tehokkainta moniosaisen indeksin käyttötapaa. Katsotaan taas esimerkkiä.

```
SELECT liike_ID
FROM myynti
WHERE aika_ID = 23
AND tuote_ID = 3400
```

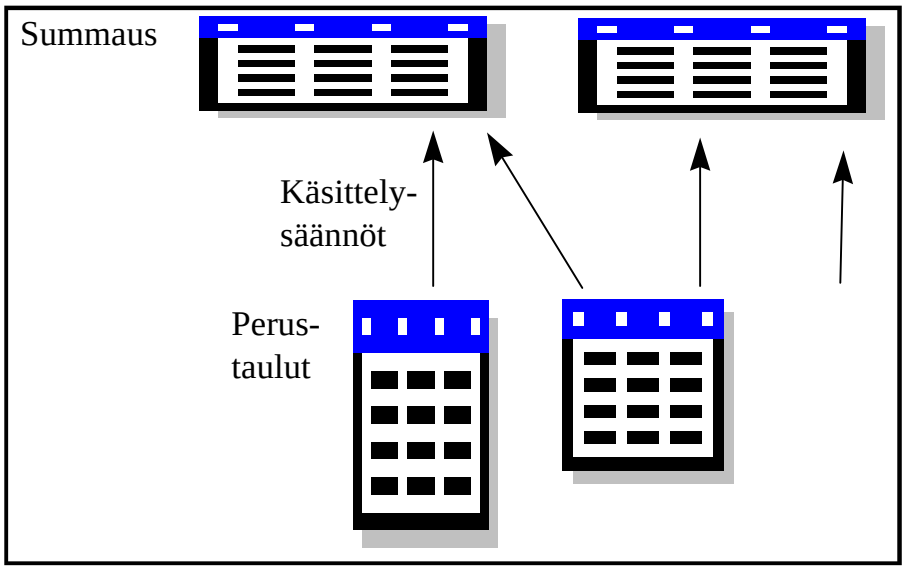
Eli haemme liike_ID:n. Nyt WHERE - lauseen hakuehdot löytyvät indeksin alkuosta, jolloin indeksi “puree” hyvin (vrt. sukunimellä haku puhelinluettelosta). Itse haettava sarake on sekin indeksissä. Koko haku voidaan siis tehdä pelkästä indeksistä, itse tauluun ei tarvitse mennä lainkaan. Tätä kutsutaan englannin kielellä termillä index-only - haku. Haku on erityisen tehokas, sillä halutut rivit ovat peräkkäin indeksissä, jolloin ne saadaan nopealla limittyvällä peräkkäisluvulla ulos. Todennäköisesti tulee yksi ainoa haja-I/O indeksiin eli hakuaika on vain 20 ms.

Summaus

Summauksen idea on laskea ja tallettaa valmiiksi käyttäjien usein kyselemiä summia. Miksi laskea aina uudestaan ja uudestaan samoja lukuja? Tässä hyödynnetään myös eräajoja: lasketaan latauksen jälkeen yöaikana summia valmiiksi ja nopeutetaan siten hakuja. Seuraavalla sivulla oleva kaavio kuvaa summauksen periaatetta.

Paksu indeksi esimerkkikyselyssä

Seuraavaksi tarkastellaan miten moniosainen, paksu indeksi voisi nopeuttaa esimerkkikyselyämme. Paksu indeksi muodostetaan lisäämällä aiemmin esittelemääni, jo olemassaolevaan Myynti-taulun indeksiin (aika_ID, tuote_ID, liike_ID) lisää sarakkeita. Loppuun laitetaan sarakkeet myynti_mk ja kulut_mk, saamme siis indeksin (aika_ID, tuote_ID,



Summaamisella saavutetaan monia etua :

1. **Kyseltävä massa pienenee**, koska summataulussa on huomattavasti vähemmän rivejä kuin perustauluissa. CPU ja I/O-kuorma pienenevät, jolloin kyselyt nopeutuvat, yleensä huomattavasti.
2. Summataulut luodaan usein **monesta taulusta** liitoksella, siis denormalisoiden. Kun summatauluun on valmiiksi koottu monen taulun tietoja, yksinkertaistuvat kyselyt huomattavasti ja myös nopeutuvat.
3. Yleisesti tarvittavat, perustiedoista **johdetut tiedot** lasketaan summaamisen yhteydessä ja ne sijoitetaan valmiiksi laskettuina summatauluihin. Kun johtamiskaavasta on sovittu, on kaikilla nyt samat keskeiset tunnusluvut käyttävissä. Käyttäjien ei tarvitse aina uudestaan laskea samoja lukuja rasittaen suorituskykyä.
4. **Monimutkaiset summaamiset** tehdään keskitetysti ja oikein (käyttäjä ei pääse tekemään itse outoja yhdistelmiä).
5. Päästään hyödyntämään helppokäyttöisiä ja voimakkaita raporttigeneraattoreita ja kyselyvälineitä; parhaimmillaan haut tulevat yhdestä taulusta (kts. kohta 2) eikä raporttiohjelmassa enää tarvita juurikaan laskentakaavoja tai käsittelylogiikkaa (kts. kohta 3), lähinnä vain hakuehtoja.

liike_ID, myynti_mk, kulut_mk). Näin muodostimme moniosaisesta indeksistä paksun indeksin lisämäällä luettavaa tietoa hakusarakkeiden perään, tavoitteena siis lukea tehokkaasti **vain** indeksiä menemättä itse tauluun ollenkaan.

Mitä haittaa tällaisesta indeksin “lihottamisesta” on? Tietysti levytilaa kuluu enemmän. Jos tiedot ovat tietotyypiltään desimaalilukuja ja 10 merkkiä pitkiä, on tarvittava lisätila noin 20 gigatavua (1 000 000 riviä). Toisin sanoen kokonaislevytarpeemme kasvaisi lähes 20% tästä yhdestä ainoasta indeksistä - vaikkakaan 20 GB ei enää ole kustannusongelma monissa laiteympäristöissä.

Toisena paksun indeksin haittana epäillään yleensä päivityksen hidastumista. Tähän faktatauluun ladataan päivittäin noin miljoona riviä eli päivän tapahtumat (tietokantajärestelmän lataaja-ohjelmalla tai itse ohjelmoiduilla INSERT-käskyillä). Faktatauluun indeksit muodostetaan latauksessa tai erikseen latauksen jälkeen. Lisäyksen I/O-määrä ei kuitenkaan tässä tapauksessa hämmästyttävää kyllä juurikaan lisäänny, sillä indeksirivin lisäys vaatii yleensä yhden luku- ja yhden kirjoitus-I/O:n riippumatta sarakkeiden määrästä. I/O tulee kuitenkin, oli indeksissä sitten yksi vai kolme saraketta.

Tarvitsemme 5 % päivistä. Indeksin kooksi voi tulla n. 45 GB paksunnettuna. Indeksiä luetaan siis 5% 45 gigasta eli 2 gigaa. Tuotteista luetaan 0, 5% eli noin 50. Ajan sisällä tuotteet eivät ole täysin peräkkäin vaan kahtena viipaleena vastaten kahta haetttavaa kuukautta. Nämä kaksi viipaletta on käytävä läpi. Saadaan 2GB * 500 s/GB eli 1000 s. eli n. 15 minuuttia. Aika hyvä aika siis. Ilman moniosaista indeksiä tulisi tauluun lukuja vielä 0, 5 % viidestä prosentista eli 250 000 * 20 ms. Tämä tekee 5 000 s eli 1 h 15 min. Paksu indeksi säästää aikaa noin tunnin.

Paksun indeksin etu siis on, että itse faktatauluun eí tarvitse mennä lainkaan. Kaikki tiedot löytyvät indeksistä.

Summataulu esimerkkikyselyssä

Esimerkkikyselyä nopeuttava summataulu pitää sisällään sarakkeet kuukausi, tuoteryhmä, liike_ID, myynti_mk_summa ja kulut_mk_summa. Summataulu rakennetaan seuraavalla SQL - lauseella:

```
INSERT INTO liike_summa
            (kk,
            tuoteryhma,
            liike_ID,
            myynti_mk_summa,
            kulut_mk_summa,
            tuotto_summa)
SELECT      aika.kk,
            tuote.tuoteryhma,
            liike.liike_ID,
            sum(myynti_mk),
            sum(kulut_mk),
            sum(myynti_mk -
            kulut_mk)
FROM
aika, myynti, tuote
WHERE
aika.aika_ID = myynti.aika_ID
AND
tuote.tuote_ID = myynti.tuote_ID
GROUP BY
aika.kk, tuote.tuoteryhma,
liike.liike_ID
```

Hakukysely puolestaan olisi summataulusta nyt tällainen:

```
SELECT    liike_ID, myynti_mk_summa, kulut_mk_summa, tuotto_summa
FROM      liike_summa
WHERE     tuoteryhma = : tuoteryhma
AND       kuukausi in (:X,:Y)
```

Kysely siis myös yksinkertaistuu huomattavasti.. Tarkastellaanpa stten vielä summataulun haittoja. Levytilaa tietysti kuluu. Tuotteita on 0,5 % taulusta ja tuoteryhmiä on 200. Summataulun rivin pituus on 40 tavua. Tämän summataulun kooksi tulee 36 (kk) * 1000 (liike) * 200 (tuoteryhma) * pituus 40 merkkiä. Jos kaikkia kombinaatioita ei ole, voi kooksi tulla esim. vain 150 MB. Siis lisätilaa tarvitaan kuitenkin hyvin vähän näin tarkasti kyselyyn sovitetulla summataululla.

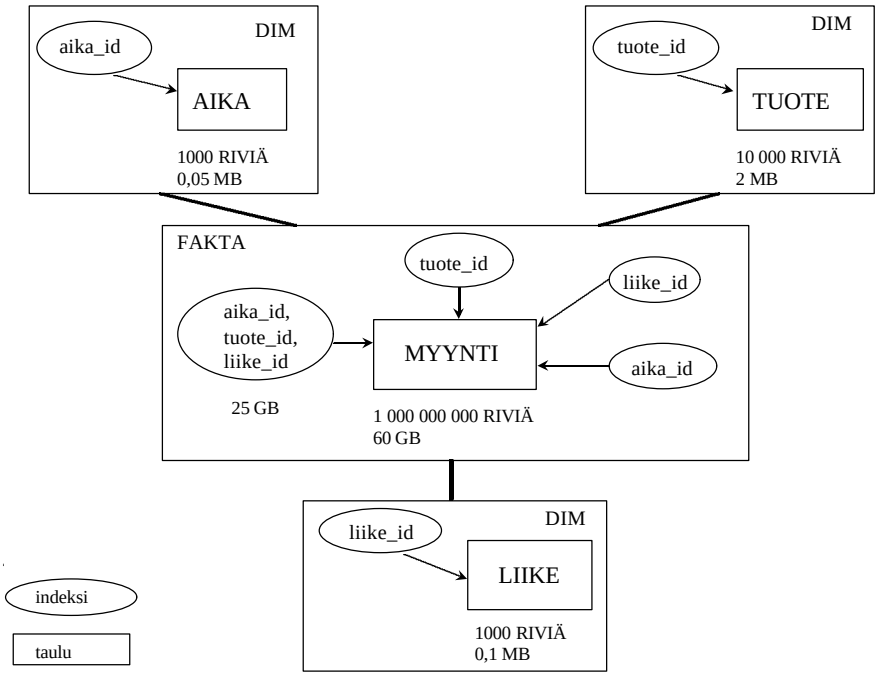
Toinen haitta on se, että summataulun luonti vie aikaa. Tässä tapauksessa summataulun luonti veisi noin 1,5 h, kuten edellisessä kappaleeessa laskettiin. Summataulua tosin ladattaisiin vain kerran kuussa. Summataulujen käyttäminen tietysti monimutkaistaa koko tietovarastoympäristöä, mitä voi pitää myös haittana. Tietojen eheydestä on on oltava tarkkana. On huolehdittava, että summataulut varmasti ovat samalla tasolla kuin perustaulutkin.

Tarkastellaan sitten minkälaiset hakuajat saadaan summataulusta. Tuote_ID -indeksistä luetaan 5% ja vastavasti taulusta 5 %. Summataulun indeksin lukuaika on niin pieni, ettei sitä tarvitse ottaa huomioon. Taululuvusta (siis 5%) saadaan 150 MB * 0,5 s/MB eli noin 10s.

Yhteenveto

Kun taulun läpiluku kestää kahdeksan tuntia, saamme paksulla indeksillä tuloksen noin 15 minuutissa. Levytilatarve ei tosin ole aivan pieni, eli suunnilleen. 20% lisätilatarvetta. Ehdottomasti paras tulos esimerkkikyselylle saadaan summataululla, jonka avulla pääsemme arviolta 10 s hakuun.

Tuotteiden optimoijat alkavat olla “summataulutietoisia”. Optimoija huomaa, että perustauluun kohdistuvan kyselyn saakin nopeammin tehtyä summataulusta ja ohjaa kyselyn läpinäkyvästi summatauluun.



Haluamme tuottaa isosta MYYNTI faktataulusta tietyn tuoteryhmän liikekohtaisen tuottoraportin, jossa vertaamme kahden eri kuukauden myyntiä.

```
SELECT   LIIKE_NIMI,
         KUUKAUSI
         SUM(MYYNTI_MK),
         SUM(KULUT_MK),
         SUM(MYYNTI_MK - KULUT_MK) AS TUOTTO
FROM     AIKA, TUOTE, MYYNTI, LIIKE
WHERE    AIKA.AIKA_ID = MYYNTI.AIKA_ID
         AND TUOTE.TUOTE_ID = MYYNTI.TUOTE_ID
         AND LIIKE.LIIKE_ID = MYYNTI.LIIKE_ID
         AND TUOTERYHMÄ = :TUOTERYHMÄ
         AND KUUKAUSI IN (:X, :Y)
GROUP BY LIIKE_NIMI, KUUKAUSI
```

Oletuksia:
kiinnostavaan tuoteryhmään kuuluu 0,5 % tuotteista
myyntitaulun rivit ovat päivätasoisia ja kaksi kuukautta edustaa 5 % päivistä

Dimensiotaulun AIKA läpiluku kestää alle 0,5 s => 0 s
Dimensiotaulun TUOTE läpiluku kestää alle 0,5 s => 0 s

Kuva: Myyntifaktat ja dimensiot ja Myyntituottokysely

Ari Hovi,
Ari Hovi Oy
ari@hovi.pp.fi