Datakuutio

Jan Lindström

Helsinki 7. marraskuuta 1997

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

Tietämyksen muodostamisen seminaari

Sisältö

1 Johdanto

Päätöksentukijärjestelmät ovat tärkeä tietokantojen sovellusalue. Yritykset keräävät tietoa erilaisista operaatioista. Tiedot talletetaan keskustietokantaan, jota yleisesti kutsutaan *tietovarastoksi* (data warehouse). Yritykset käyttävät kerättyä operatiivista tietoa helpottamaan liiketoiminnan ymmärtämistä ja hoitamista.

Analysoijat käyttävät tietovarastoa liiketoimintatietojen poimintaan. Poimittuja tietoja käytetään päätöksentekoon. Tätä interaktiivista päätöksentekoprosessia kutsutaan *OLAP*:ksi (Online Analytical Processing) [CCS93]. OLAP ohjelmistoissa tieto esitetään useista liiketoiminnan näkökulmista (ulottuvuuksista). Kyselyiden suorittamiseksi ja tehostamiseksi useista näkökulmista samanaikaisesti tarvitaan uusia menetelmiä ja käsitteitä.

Tämän esityksen tarkoituksena on esitellä tarvittavat käsitteet ja menetelmät useampiulotteisten interaktiivisten kyselyiden toteuttamiseen relaatiotietokannoista. Kappale 2 esittelee ongelman ja ongelmaan kehitetyn ratkaisun käsitteet. Kappale 3 esittelee kuinka tarvittavat tiedot lasketaan relaatiotietokannasta. Kappaleessä 4 käsitellään yksi laskentatekniikka yksityiskohtaisesti. Lopuksi esitetään yhteenveto.

2 Datakuutio

Relaatiotietokannat mallintavat N-ulotteista tietoa relaatiossa, jossa on N-attribuuttia. Esimerkiksi 4-ulotteinen lämpötilatieto esitetään kuusisarakkeisessa relaatiossa (taulukko 1). Neljä ensimmäistä saraketta kuvaavat neljää ulottuvuutta x, y, z, $aika$. Loput sarakkeet kuvaavat mittauksia neliulotteisessa tilassa.

Taulukko 1: Säätietoja.

Aika	Leveys	Pituus	Korkeus	Lämpö	Paine
27-01-1994 15:00	37.84 N	12.45 W	10	21	1009
27-11-1994 15:00	34.16 N	27.05 W	10	23	1024

SQL-standardi tarjoaa viisi koostefunktiota. Funktiot ovat $COUNT$, SUM, MIN, MAX ja AVG. Esimerkkinä koostefunktion käytöstä on lämpötilojen keskiarvon laskenta edellisestä aineistosta (kuva 1).

```
SELECT AVG(Lämpö)

FROM Säätaulu;
```

Kuva 1: Lämpötilojen keskiarvo.

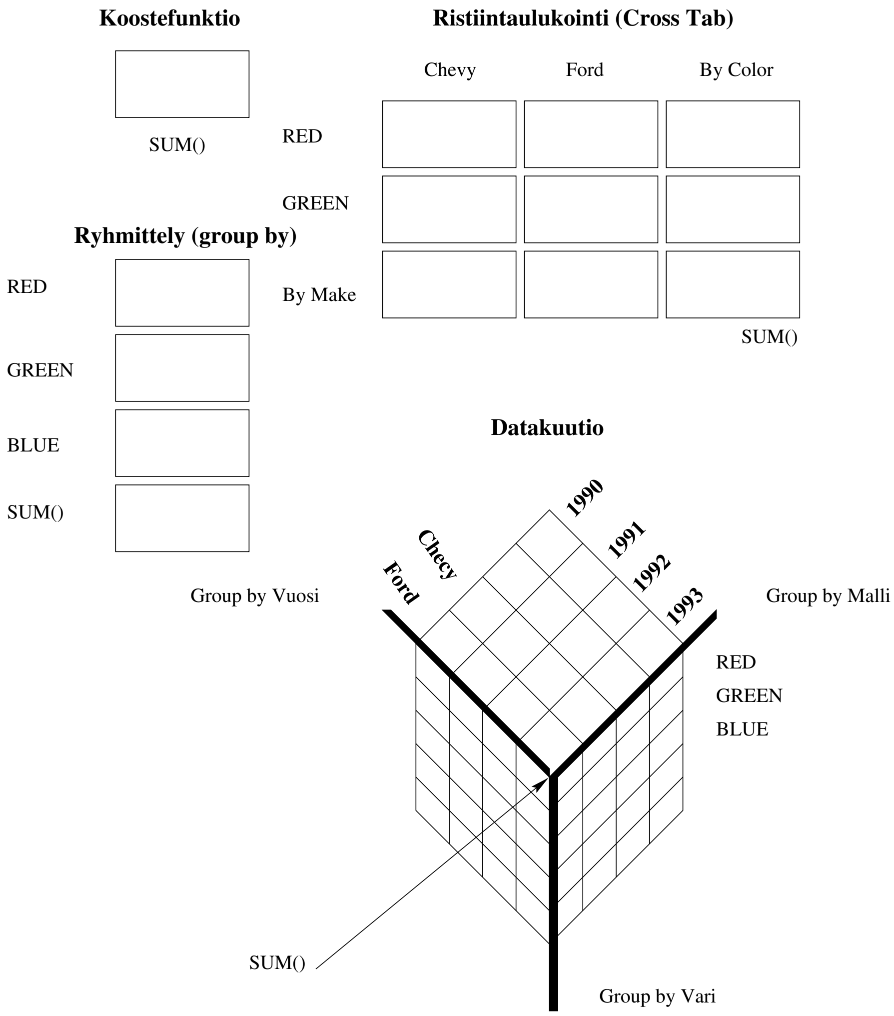
Koostefunktiot palauttavat yksittäisen arvon. Käyttämällä GROUP BY -operaattoria voidaan muodostaa relaatio koostearvoista. GROUP BY -operaattori jakaa taulun ryhmiin ryhmittelyattribuuttien arvojen mukaisesti. Jokaiselle ryhmälle suoritetaan koostefunktio. Esimerkissä 2 keskimääräinen lämpötila ryhmiteltynä ajan ja korkeuden mukaisesti.

```
SELECT Aika, Korkeus, AVG(Lämpö)
FROM Säätaulu
GROUP BY Aika, Korkeus;
```

Kuva 2: Keskimääräiset lämpötilat.

Liiketoiminnan tietoanalyyseissa on tarpeen ryhmitellä tietoa useiden attribuuttien mukaisesti. Vastausajan lyhentäminen on ensisijainen tavoite tällaisissä kyselyissä. Interaktiivinen analyysi ei ole mahdollista, jos jokainen koostefunktio lasketaan suoritushetkellä. Useimmat valmiit ohjelmistot laskevat koostefunktiot valmiiksi ja suorittavat interaktiiviset kyselyt valmiiksi laskettuun tietoon.

Datakuutio (Data Cube, CUBE) [GBLP96] on n-ulotteinen yleistys *ryhmittelyoperaattorista* (group-by operator) (kuva 3). Datakuutio n-ulottuvuudessa vastaa joukkoa ryhmittelylauseita, joista jokainen on osajoukko datakuutiosta. Datakuutio on eräänlainen useampiulotteinen näkymä tietokantaan, missä kriittinen arvo kuten myynti on organisoitu useisiin ulottuvuuksiin. Esimerkiksi kuvassa 3 autojen myynti on organisoitu mallin, vuoden ja värin mukaisesti kolmiulotteiseksi kuutioksi. Jokainen solu datakuutiossa vastaa yksikäsitteistä arvojoukkoa eri ulottuvuuksissa ja sisältää mitatun arvon.



Kuva 3: Datakuutio.

Datakuutio voidaan esittää lauseella:

CUBE [DISTINCT | ALL] <valintalista> BY <koostelista> <taululause>

Datakuutio laskee koostefunktiot *koostelistasta* (aggregate list) ryhmittelemällä kaikki mahdolliset osajoukot attribuuteista, jotka ovat *valintalistassa* (select list). Termillä *kuutio* (cuboid) tarkoitetaan ryhmittelykoostetta. Ryhmittelykoostetta kaikista attribuuteista valintalistalta kutsutaan *pohjakuutioksi* (base cuboid). Datakuutio n:lle attribuutille aiheuttaa 2^n kuution ja yhden pohjakuution laskennan.

Esimerkissä 4 lasketaan myynnillä ryhmiteltynä kuutiot kaikille kahdeksalle osajoukolle joukosta {Tuote,Vuosi,Asiakas} mukaanlukien tyhjä osajoukko

```
CUBE Tuote, Vuosi, Asiakas BY SUM(Myynti);
```

Kuva 4: Myyntikuutio.

Datakuutioon voidaan tehdä erilaisia kyselyitä. Kyselyitä ovat esimerkiksi osavälikyselyt [HAMS97] ja *osasummat* (partial-sum) [HBA97]. Tässä esityksessä keskitymme datakuution laskentaan.

3 Datakuution laskentamenetelmiä

Datakuution laskentatekniikkaan vaikuttaa koostefunktio. Oletamme, että koostefunktio on *jakautuva* (distributive). Esimerkiksi funktiot sum ja min ovat jakautuvia. Jakautuvilla funktioilla ylijoukko voidaan laskea alijoukosta. Esimerkiksi jos $Y \cap Z = \emptyset$ ja $X = Y \cup Z$,niin tällöin $sum(X) = sum(Y) + sum(Z)$.

Suoraviivainen lähestymistapa on laskea riippumattomasti jokainen kuutio pohjakuutiosta käyttäen ryhmittelytekniikoita. Pohjakuutio luetaan levyltä ja sen sisältö läpikäydään jokaisen kuution laskennassa. Tällainen laskentatapa on suurilla datakuutioilla tehotonta.

Datakuution laskennassa lasketaan kaikki kuutiot jotka muodostavat datakuution. Pohjakuutio lasketaan alkuperäisestä relaatiosta. Pohjakuution laskenta voidaan suorittaa käyttäen standardeja ryhmittelymenetelmiä kuten lajittelua ja hajautusta. Muut kuutiot voidaan laskea pohjakuutiosta.

Esimerkiksi myyntiohjelmiston relaatiossa myynnin määrä tuotteittain ja asiakkaittain voidaan laskea käyttäen ryhmittelyä myynnin määrä tuotteittain, vuosittain ja asiakkaittain. On valittavana useita mahdollisiä järjestyksiä kuutioiden laskentaan ja useita mahdollisia menetelmiä ryhmittelyn laskentaan. Näiden erilaiset kombinaatiot tuottavat erilaisia tekniikoita datakuution laskentaan. Seuraavaksi esittelemme erilaisia datakuution laskentatekniikoita lyhyesti, jonka jälkeen esitämme yhden tehokkaimmista laskentatekniikoista yksityiskohtaisemmin.

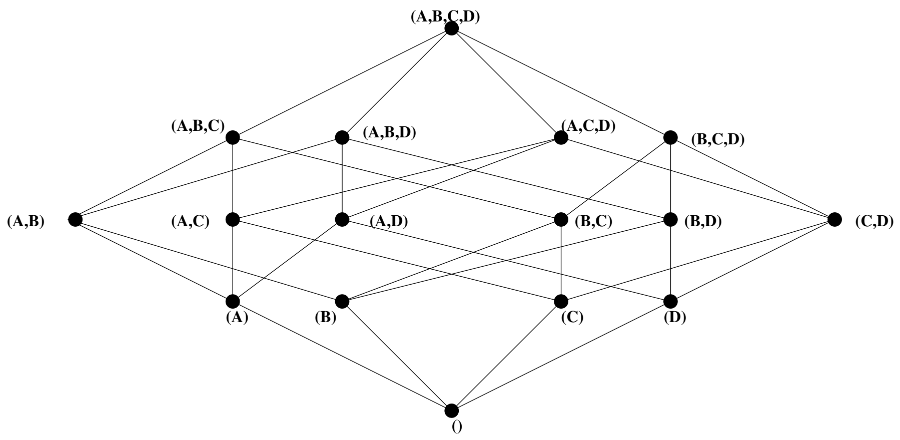
Olkoon k:n attribuutin laskettava datakuutio $X = \{A_1, A_2, ..., A_k\}$ relaatiosta R käyttäen koostefunktiota $F()$. Kuutio $S = \{A_{i_1}, A_{i_2}, ..., A_{i_j}\}$ j:lle attribuutille voidaan määritellä seuraavasti:

$$\begin{aligned} CUBOID(S) = \{(a_{i_1}, a_{i_2}, ..., a_{i_j}, F(\tau)) | \tau \subseteq R, t \in \tau \iff \\ \prod A_{i_1}, A_{i_2}, ..., A_{i_j}(t) = (a_{i_1}, a_{i_2}, ..., a_{i_j})\} \end{aligned} \tag{1}$$

Epäformaalisti kuutio on ryhmittely attribuuteille $A_{i_1}, A_{i_2}, ..., A_{i_j}$ käyttäen koostefunktiota F. Tällainen kuutio voidaan esittää $k+1$ attribuutin relaatiolla käyttäen erityisarvoa ALL lopuille $k - j$ attribuutille [GBLP96].

Tarkastellaan kuutioiden laskentaa datakuutiosta attribuuteilla $\{A, B, C, D\}$. Kuutio $\{A, C\}$ voidaan laskea kuutiosta $\{A, B, C\}$, koska koostefunktio on jakautuva. Yleisesti kuutio attribuuttijoukolla X voidaan laskea kuutiosta attribuuttijoukolla $Y \iff X \subset Y$. Kuutio $k - 1$:llä attribuutilla laskentaan kuutiosta k:lla attribuutilla käytetään heuristiikkaa, koska kuution koko yleensä suurenee samalla kun attribuuttien määrä kasvaa. Esimerkiksi on parempi laskea myynnin summat tuotteittain käyttäen ryhmittelyä myynnin summat tuotteittain ja asiakkaittain kuin käyttämällä ryhmittelyä myynnin summat tuotteittain, vuosittain ja asiakkaittain.

Tällainen laskentahierarkia voidaan esittää suunnattuna syklittömänä verkkona (DAG), missä solmut ovat kuutioita ja verkossa on kaari $A \to B \iff B \subseteq A$. Esimerkiksi datakuution $\{A, B, C, D\}$ laskentahierarkia on esitetty kuvassa 5.



Kuva 5: Laskentahierarkia.

Vanhempimenetelmässä (parent method) [DANR96] jokainen kuutio lasketaan yhdestä kuution vanhemmista verkossa. Tämä menetelmä on tehokkaampi kuin riippumaton menetelmä, koska vanhempi on yleisesti pienempi kuin pohjakuutio.

Limitysmenetelmä (overlap method) [DANR96, AAD$^+$96] on laajennus vanhempi menetelmään. Vanhempimenetelmän tavoin limitysmenetelmässä kukin kuutio lasketaan yhdestä vanhemmasta kuutiopuussa. Vanhempi menetelmää pyritään tehostamaan limittämällä eri kuutioiden laskentaa. Tämä voi ratkaisevasti vähentää levyoperaatioiden määrää.

Datakuutioiden laskennan apuna voidaan käyttää myös osittain valmiiksi laskettuja kuutioita [HRU96] ja indeksejä [GHRU97]. Tässä esityksessä keskitymme limitysmenetelmään.

4 Limitysmenetelmä

Tarkastellaan kuutiota $\{A_1, A_2, ..., A_j\}$ j:llä attribuutilla. Merkinnällä $(A_1, A_2, ..., A_j)$ tarkoitetaan kuutiota lajiteltuna attribuuttien $A_1, A_2, ..., A_j$ mukaiseen järjestykseen. Tarkastellaan kuution $S = (A_1, A_2, ..., A_{l-1}, A_{l+1}, ..., A_j)$ laskentaa käyttäen kuutiota $B = (A_1, A_2, ..., A_j)$. *Luokitteluajo* (sorted run) R S:stä käyttäen B:tä määritellään seuraavasti:

$R = \prod A_1, A_2, ..., A_{l-1}, A_{l+1}, ..., A_j(Q),$

missä Q on maksimaalinen ketju rivejä τ B:ssä siten, että jokaiselle riville Q:ssa l:ssä ensimmäisessä sarakkeessa on sama arvo. Epäformaalisti luokitteluajo S:stä käyttäen B:tä on maksimaalinen määrä rivejä B:ssä, joiden järjestys on sama kuin rivien järjestys S:ään liittyvässä lajittelujärjestyksessä.

Esimerkiksi tarkastellaan B = [(1,1,2), (1,1,3), (1,2,2), (2,1,3), (2,3,2), (3,3,1)]. Olkoon laskettava kuutio S ensimmäisen ja kolmannen attribuutin mukainen. Eli S = [(1,2), (1,3), (2,3), (2,2), (3,1)]. Luokitteluajot S:lle ovat [(1,2)), (1,3)], [(1,2)], [(2,3)], [(2,2))] ja [(3,1)].

Kuution S *ositus* (partition) B:ssä on yhdiste luokitteluajoista, joissa ensimmäisillä $l - 1$ sarakkeella kaikista riveistä on sama arvo. Edellisessä esimerkissä ositukset S:lle B:ssä ovat [(1,2), (1,3)], [(2,2), (2,3)] ja [(3,1)].

Limitysalgoritmi aloittaa lajittelemalla pohjakuution. Kaikki muut kuutiot voidaan suoraan laskea luokitteluajoilla ilman lisäjärjestämistä. Oletetaan pohjakuution datakuution $\{A, B, C, D\}$ olevan järjestetty järjestykseen (A, B, C, D). Pohjakuution lajittelujärjestys määrää muiden kuutioiden laskennan. Luokitteluajot muille kuutioille kuutiosta $\{A, B, C, D\}$ on esitetty kuvassa 5.

Jokaisen kuution laskenta vaatii saman määrän muistia. Jos muistia on riittävästi kaikkien kuutioiden laskentaan, koko datakuutio voidaan laskea yhdellä syöterelaation läpikäynnillä. Yleisesti muisti ei kuitenkaan riitä suurien datakuutioiden laskentaan. Maksimaalisen kuutioiden laskennan limittymisen saavuttamiseksi pyritään vähentämään yksittäisen kuution laskentaan tarvittavaa muistimäärää. Ideana on, että kuutiota laskettaessa toisesta lajitellusta kuutiosta tarvitaan muistia säilyttämään vain osa kuutiosta.

Osan laskennan valmistuttua rivit *kanavoidaan* (pipelined) lapsitason kuutioiden laskentaan tai kirjoitetaan levylle. Samaa muistialuetta voidaan käyttää seuraavan osion laskennassa. Esimerkiksi laskettaessa kuutioita (A, B, C) ja (A, B, D) kuutiosta (A, B, C, D) on osituksen koko kuutiolle (A, B, C) 1 kun osituksen koko kuutiolla (A, B, D) on rajoitettu D:n eroavien arvojen lukumäärään. Muistitila näiden osien säilyttämiseen on riittävää laskentaan. Joten useiden kuutioiden laskenta voidaan limittää vapaana olevaan muistiin tehokkaasti vähentäen läpikäyntien määrää. Kuutioiden laskennan limittämisessä on kolme tarkasteltävaa kohtaa:

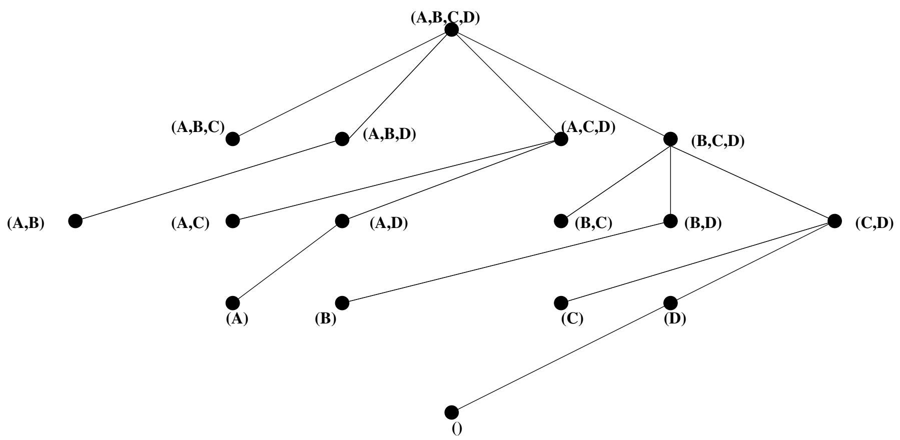
1. vanhemman valinta kuution laskemiseksi,

2. kuution laskenta vanhemmasta ja

3. kuutiojoukkojen valinta limittyvään laskentaan.

4.1 Vanhemman valinta

Jokaisella kuutiolla verkossa on useampi kuin yksi vanhempi, josta kuutio voidaan laskea. Päättämällä mitä vanhemmista käytetään verkko muuttuu puuksi. Juuressa on pohjakuutio ja jokaisella kuutiolla on vanhempi. Esimerkiksi datakuution (A, B, C, D) yksi mahdollinen laskentapuu on esitetty kuvassa 6.



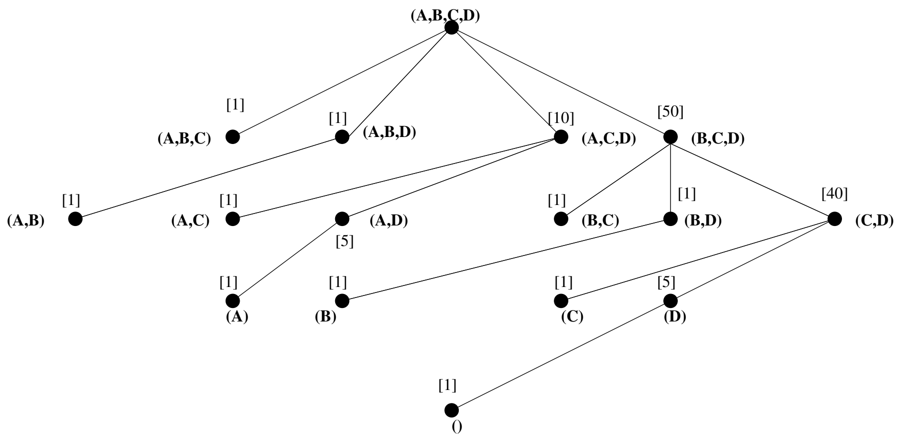
Kuva 6: Laskentapuu.

Useita muita mahdollisia puita on olemassa. Tavoitteena puun valinnassa on minimoida kuutioiden ositusten kokoa, jotta kuution laskentaan tarvittaisiin mahdollisimman vähän muistia. Esimerkiksi on tehokkaampaa laskea kuutio (A, C) kuutiosta (A, C, D) kuin kuutiosta (A, B, C), koska kuution (A, C, D) lajittelujärjestys on sama kuin kuutiossa (A, C) ja osituksen koko on 1.

Osien koko voidaan arvioida kuutioiden kokojen arvioista. Oletetaan että jokin laskelma tarjoaa kaikkien datakuution kuutioiden koot. Jos kuutioiden kokotietoa ei ole olemassa, voidaan koot arvioida käyttäen tasaista arvojen jakautumisoletusta jokaisen datakuution attribuutin erilaiseen arvoon suhteutettuna ja pohjakuution kokoa. Kuution $(A_1, A_2, ..., A_{l-1}, A_{l+1}, ..., A_k)$ osan koko laskettuna kuutiosta $(A_1, A_2, ..., A_k)$ on arviolta yhtä suuri kuin:

$$\frac{\text{Kuution } (A_1, A_2, ..., A_{l-1}, A_{l+1}, ..., A_k) \text{ koko}}{\text{Kuution } (A_1, A_2, ..., A_k) \text{ koko}}$$

Tarkastellaan kuutiota $S = (A_{i_1}, A_{i_2}, ..., A_{i_j})$, missä pohjakuutio on $(A_1, A_2, ..., A_k)$. Kuutio S voidaan laskea mistä kuutiosta tahansa yhdellä lisäattribuutilla, jota merkitään A_l. Heuristiikkana on valita kuutio, jolla on suurin l:n arvo kuution S laskentaan. Laskentapuu kuvassa 7 on muodostettu käyttäen tätä heuristiikkaa. Laskentapuuhun osien sivuarviot on merkitty merkinnällä [n]. Esimerkiksi osituksen koko kuution (A, B, C) laskemiseksi kuutiosta (A, B, C, D) on 1 kun osituksen koko kuutiolle (B, C, D) on sama kuin kuution koko.



Kuva 7: Laskentapuu kokoarvioineen.

4.2 Kuution laskenta vanhemmasta

Jokainen muu kuutio S paitsi pohjakuutio voidaan laskea vanhemmasta B laskentapuusta. Jos muistitila riittää suurimman S:n osion laskentaan B:stä, voidaan koko kuutio S laskea yhdellä B:n läpikäynnillä. Jos muistitila ei riitä, kirjoitetaan levylle S:n luokitteluajot ensimmäisellä kierroksella ja tämän jälkeen yhdistetään luokitteluajot lopullisen kuution laskemiseksi myöhemmillä kierroksilla (kuva 8).

```
Syöte: Muistitila M varattuna tälle
laskennalle ja laskennan tila.
Tulos: Lajiteltu kuutio S.

for ∀ τ ∈ B
{
    if ( state == Partition) then
        process_partition(τ);
    else  /* state == SortRun */
        process_sorted_run(τ);

    end_of_cuboid();
}
```

Kuva 8: Kuution laskenta-algoritmi.

Proseduuri process_partion() suorittaa seuraavat toimenpiteet:

- Jos syöterivi aloittaa uuden osan, tulosta käsittelyssä oleva osa kuution lopussa ja aloita uudesta osiosta.

- Jos syöterivi on sama kuin osassa jo oleva rivi, uudelleenlaske koostefunktio osassa olevalle riville käyttäen vanhaa koostearvoa ja syöteriviä.

- Jos syöterivi ei ole sama kuin osassa oleva rivi, lisää syöterivi käsittelyssä olevaan osaan oikealle paikalleen säilyttäen lajittelujärjestys osassa.

Oletetaan, että muistitila on suurempi kuin suurimman osion koko. Myöhemmin esitetään kuinka hallitaan tilanne, jossa osio ei mahdu muistiin dynaamisesti varaamalla sivuja. Proseduuri process_sort_run() suorittaa seuraavat toimenpiteet:

- Jos syöterivi aloittaa uuden luokitteluajon, kirjoita kaikki käsittelyssä olevan luokitteluajon sivut levylle ja aloita uusi luokitteluajo.

- Jos syöterivi on sama kuin viimeinen rivi luokitteluajossa, uudelleenlaske viimeisen rivin koostefunktio käyttäen vanhaa koostearvoa ja syöteriviä.

- Jos syöterivi ei ole sama kuin viimeinen rivi luokitteluajossa, lisää syöterivi luokitteluajon loppuun. Jos muistitila ei riitä syöterivin lisäykseen, kirjoita kaikki luokitteluajon sivut levylle ja jatka luokitteluajoa syöterivillä.

Proseduuri end_of_cuboid() kirjoittaa lopullisen osion tai muistissa olevan luokitteluajon levylle. *Partition* tilassa kuutiot lasketaan kokonaisuudessaan ensimmäisellä läpikäynnillä. *SortRun* tilassa levyllä on joukko luokitteluajoja. Jokainen kuutio lasketaan yhdistämällä luokitteluajot yhdistäen duplikaatit koostefunktiolla.

4.3 Kuutiojoukon valinta

Tarkoituksena on valita kuutiojoukko, joka voidaan laskea samanaikaisesti annetulla muistimäärällä. Osa kuutioista saattaa olla *Partition* tilassa ja vaatia osion koon verran muistia. Loput kuutiot ovat *SortRun* tilassa ja tarvitsevat yhden sivun muistia. Oletamme että kuutioiden kokojen arviot ovat olemassa. Kuutioiden koon arviointia esitellään artikkelissa [SDNR96].

Valitussa alipuussa laskentapuussa ja vapaalle muistimäärälle M merkitään laskettavat kuutiot ja varataan kuutioille muistitila. Jos kuutio on *Partition* tilassa, kuution rivit voidaan kanavoida lapsikuutioiden laskentaan samalla läpikäynnillä. Samaa menetelmää ei voida käyttää *SortRun* tilassa. *SortRun* tilassa on kaksi rajoitusta:

1. Kuutio voidaan harkita laskettavaksi jos kuution vanhempi on alipuun juuri tai jos vanhempi on *Partition* tilassa.

2. Varattu kokonaismuistitila kaikille kuutioille ei saa ylittää arvoa M.

5 Yhteenveto

Päätöksentukijärjestelmät ovat tärkeä tietokantojen sovellusalue. Yritykset käyttävät kerättyjä operatiivisia tietoja helpottamaan liiketoiminnan ymmärtämistä ja hoitamista. Tähän tarkoitukseen tietoa yrityksessä kerätään erilaisista operaatioista ja kerätyt tiedot talletetaan tietovarastoon.

Tietovarastoa käytetään liiketoimintatietojen poimintaan. Poimittujen tietojen avulla tehdään päätöksiä. Tätä interaktiivista päätöksentekoprosessia kutsutaan OLAP:ksi. OLAP ohjelmistoissa tieto esitetään useista liiketoiminnan näkökulmista (ulottuvuuksista).

Datakuutio on useampiulotteinen näkymä tietokantaan, missä kriittinen arvo kuten myynti on organisoitu useisiin ulottuvuuksiin. Jokainen solu datakuutiossa vastaa yksikäsitteistä arvojoukkoa eri dimensioissa ja sisältää mitatun arvon.

Datakuution laskentamenetelmistä esitettiin limitysmenetelmä, joka on osoittautunut yhtä tehokkaaksi tai jopa tehokkaammaksi kuin muut esitetyt menetelmät.

Viitteet

[AAD+96] S. Agarwal, R. Agrawal, P. Deshpande, A. Gupta, J. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. In *Proc. of the 22nd Int. Conf. on Very Large Data Bases*, pages 506–521, September 1996.

[CCS93] E. F. Codd, S. B. Codd, and C. T. Salley. Providing olap (on-line analytical processing) to user-analysts: An it mandate. Tech. report, E. F. Codd and Associates, 1993.

[DANR96] P. Deshpande, S. Agarwal, J. Naughton, and R. Ramakrishnan. Computation of multidimensional aggregates. Tech. Report CS-TR-96-1314, University of Wisconsin, Madison, May 1996.

[GBLP96] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In *Proc. of the 12th Int. Conf. on Data Engineering*, pages 152–159, February 1996.

[GHRU97] H. Gupta, V. Harinarayan, A. Rajaraman, and J. Ullman. Index selection for olap. In *Proc. of the 13th Int. Conf. on Data Engineering*, Birmingham, U.K., April 1997.

[HAMS97] C. Ho, R. Agrawal, N. Megiddo, and R. Srikant. Range queries in olap data cubes. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 73–88, May 1997.

[HBA97] C. Ho, J. Bruck, and R. Agrawal. Partial-sum queries in olap data cubes using covering codes. In *Proc. of the 16th ACM Symposium of Principles of Database Systems*, pages 228–237, May 1997.

[HRU96] V. Harinarayan, A. Rajaraman, and J. Ullman. Implementing data cubes efficiently. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 205–216, June 1996.

[SDNR96] A. Shukla, P. Deshpande, J. Naughton, and K. Ramasamy. Storage estimation for multidimensional aggregates in the presence of hierarchies. In *Proc. of the 22nd Conf. on Very Large Data Bases*, pages 522–531, September 1996.