Helppokäyttöisen OLAP –kyselykielen suunnittelu ja toteutus

Lasse Hirvonen

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Pro gradu -tutkielma
Maaliskuu 2001

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Lasse Hirvonen: Helppokäyttöisen OLAP –kyselykielen suunnittelu ja toteutus
Pro gradu -tutkielma, 62 sivua.

Maaliskuu 2001

Tässä tutkielmassa esitellään peruskäsitteistö OLAP (On-line Analytical Processing) –järjestelmälle ja tähän käsitteistöön perustuen luodaan perusoperaatiot moniulotteisesti organisoidun tietokuution käsittelemiseksi. Käsitteistössä erotetaan eksplisiittisesti kaavio- ja ilmentymätaso mahdollistamaan yleinen OLAP –kuution esittäminen ja siihen perustuvan kielen kehittäminen. Käyttäjäystävällisyys OLAP –kyselykieleen saadaan siten, että kieli sisältää vain kaksi korkealla abstraktiotasolla olevaa OLAP –operaatiota view ja add, joiden käyttäminen on tavalliselle loppukäyttäjälle sekä intuitiivista että deklaratiivista. Näillä operaatioilla on poikkeuksellisen suuri ilmaisuvoima. Esimerkiksi kehitetyllä view –operaatiolla on mahdollista ilmaista mikä tahansa perinteisten OLAP –järjestelmien operaatioiden kombinaatio. Täten kehitetty view –operaatio on huomattavasti korkeammalla abstraktiotasolla kuin nykyisten OLAP –kyselykielien operaatiot. Tätä piirrettä demonstroidaan tutkielmassa useiden esimerkkikyselyjen yhteydessä. Kehitettyjen operaatioiden lisäksi kieli koostuu logiikkaohjelmoinnin perusilmauksista, joita käytetään järjestelmän relationaalisesti organisoidun tiedon käsittelyyn. Kieli sisältää myös ilmaisun, jolla relationaalinen käsittely ja OLAP –käsittely integroidaan. Tekstuaalisen kyselykielen lisäksi tutkielmassa kehitetään myös graafinen kyselykieli, jonka ilmaisuvoima ei ole vielä samalla tasolla kuin tekstuaalisen kyselykielen. Kielen sisältämät operaatiot toteutettiin LPA:n Win-Prologilla. Logiikkaohjelmoinnin perusilmausten deklaratiivisuus sekä jaetun muuttujan käsite suovat mahdollisuuden esittää mutkikkaitakin OLAP –kyselyjä tiiviisti ja käyttäjäystävällisesti.

2. Johdanto

2.1. ILMIÖ

Codd [CCS93] kehitti OLAP (On-line Analytical Processing) -käsitteen määritelläkseen vaatimukset tiedonhallinnalle moniulotteisessa ympäristössä. OLAP on työkalu yhteenvetotiedon analysointiin ja visualisointiin helpottamaan loppukäyttäjän päätöksentekoa. Moniulotteisuudella OLAP-ympäristössä ymmärretään niitä tekijöitä, joiden suhteen yhteenvetotietoa on mielekästä tarkastella. Mielekkäiden ulottuvuksien määrä riippuu kohdealueesta, jolle OLAP-sovellus on kehitetty. OLAP sovellukset tarjoavat paremmat edellytykset nopeille ja vaihteleville kyselyille, koska ne heijastavat yrityksen tiedon moniulotteista luonnetta. Vielä tällä hetkellä suurin osa tietojärjestelmistä suunnitellaan OLTP (On-line Transaction Processing) –järjestelmiksi. OLTP-järjestelmät ovat monenkäyttäjän järjestelmiä, joissa analysoinnin tarve on pientä. OLAP-järjestelmiä koskevan tutkimuksen vähäisyydestä johtuu se, että niitä on toistaiseksi toteutettu melko vähän. OLAP –tutkimuksen lisääntyessä tuotetaan tulevaisuudessa yhä monipuolisempia järjestelmiä, mikä todennäköisesti merkitsee OLAP –sovellusten nopeaa lisääntymistä [Tho97, Dat00, CC93].

Tutkielman keskeisimpänä tavoitteena on kehittää helppokäyttöinen kyselykieli tietokuutiossa olevan tiedon analysointiin. Siinä oleva yhteenvetotieto on kerätty tietokannoista organisoimalla se moniulotteisesti. Tämänhetkiset OLAP –sovellukset eivät ole kyenneet täyttämään kaikkia Coddin niille määrittelemiä vaatimuksia [Tho97, Dat00, PJ99, CC93, Kos98]. Tämän tutkielman peruslähtökohtana on OLAP –järjestelmän luominen. Alusta asti on tarkoitus kiinnittää huomiota OLAP:n erityispiirteisiin sekä toisaalta eroihin OLTP:hen verrattuna. Eroja luonnehditaan esimerkiksi lähteissä [Tho97, Dat00, ZDN97, Kos98]. Tiedon kuvaamista ja perusoperaatioden luomista ja kehittämistä on tarkasteltu useissa julkaisuissa (ks. esim. [Tho97, Dat00, NJ91, AGS97, PJ99, GL97, HMV99, BPT97]). Perusoperaatioista keskeisin huomio on syytä antaa ulottuvuuksien hallinnalle sekä näkymän (view) –määrittelyn joustavuudelle. Näitä asioita tarkastellaan esimerkiksi lähteessä [BPT97]. Tutkielmassa ei rajoituta tiettyihin OLAP –sovellutuksiin kuten tiedonhaun OLAP –sovellutuksiin [MLC00, Dyr96], vaan ongelmakenttää pyritään lähestymään yleisesti. Samoin monet

OLAP:iin liittyvät erityiskysymykset kuten OLAP –kuution suunnittelu [NNT00, GL97] ja harvan tiedon kontrollointimenetelmät (controlling sparsity) [Dyr96, RS97] ovat työn ulkopuolella. Harvan tiedon ongelma tarkoittaa tilannetta, jossa suurehko osa tietokuution tietokentistä on tyhjiä. Tämä aiheuttaa liiallista tilankäyttöä ja hidastaa usein kyselyjen toteuttamista.

2.2. OLAP:N VAATIMUKSET

OLAP eroaa OLTP:stä monessa kohdin. OLTP-tietokannat on tyypillisesti suunniteltu useamman käyttäjän yhtäaikaiskäyttöön ja niissä haun kohteena on yleensä yksi tai muutama tietoalkio. OLAP-järjestelmässä sen sijaan yhtäaikaisia käyttäjiä on vain muutamia ja haun kohteena on jalostettua yhteenvetotietoa.

Nykyään tietokantojen edellytetään tukevan sekä monen käyttäjän yhtäaikaiskäyttöä että laajoja monimutkaisia kyselyjä, jotka voivat sisältää myös yhteenvetotietojen analysointia. Nykyiset taulukkolaskentapohjaiset järjestelmät eivätkä tietokantojen kyselykielet kuten SQL tue riittävästi tiedon analysointia [Tho97].

2.2.1. Nopeus

Nopeus on tärkeä OLAP:n vaatimus. Se rohkaisee tekemään yksityiskohtaisempia yhteenvetotietoja koskevia analyyseja. Jos vastausten saaminen kestää kymmeniä minuutteja on vaikeaa muistaa, mitä alunperin oltiin tarkastelemassa. Hitaus saattaa johtaa käyttäjän tekemään sellaisia kyselyjä, joiden vasteaika on lyhyempi, mutta joiden tulos ei tue päätöksentekoa riittävästi. Liiallinen hitaus siis estää tehokasta analysointia [Tho97, Kos98].

Nopeus on kuitenkin aina kompromissi kahden tarkasteltavan tekijän eli ajan ja tilan suhteen. Analysointia pystytään nopeuttamaan yksinkertaisesti laskemalla tunnuslukuja ja aggregointeja etukäteen ja tallentamalla nämä tulokset omiin tauluihinsa tiedonhaun nopeuttamiseksi. Tämä kuitenkin tarkoittaa sitä, että talletustilaa vaaditaan enemmän. Usein suurimmat ongelmat liittyvät päivitykseen. Mitä enemmän aggregointeja ja tunnuslukuja on laskettava etukäteen sitä hitaammaksi yhteenvetotietojen päivitykset käyvät. Lisäksi sovellutus on poissa käytöstä päivitysten aikana. Useilla yrityksillä ei ole varaa pitää tietojärjestelmää käyttämättömänä kovin pitkiä aikoja. Päivitysten ja nopeuden ongelmiin on

kiinnitetty suuresti huomiota, ja erilaisia tekniikoita on kehitelty näiden osa-alueiden optimoimiseksi.

Seuraavaksi esitellään lyhyesti lähestymistapoja, joilla OLAP –prosessointia pyritään nopeuttamaan. Summary-Delta Table [MQM97] kokoaa muutoksista erillisen taulun, jonka se organisoi samalla tavalla kuin todellisen tietokannan. Quasi-Cubes-systeemi [BS97] taas pilkkoo tietojärjestelmän osiksi, jonka tietoarvoille voidaan johtaa arviokaava säästäen näin tallennustilaa. Arbitrary Tiling [FB99] on systeemi, joka pilkkoo tietojärjestelmän osiin. Näiden osien koko muuttuu kyselyjen vaikutuksesta, optimoiden näin itseään. Relative Prefix Sums [CAA+99] on tekniikka, jolla luodaan muutamia älykkäästi organisoituja summatauluja summien laskemisen nopeuttamiseksi. Chunks [DRS+98] perustuu siihen, että tehokkaasti organisoituja ja sopivasti pilkottuja paloja viedään kyselyistä tietokoneen muistiin, jossa niitä voidaan käyttää sellaisenaan siinä tapauksessa, että ko. tietoa tarvitaan seuraavissa kyselyissä.

2.2.2. Ilmaisuvoima

Tietojen analyysi on paljon muutakin kuin lukujen yhteenvetojen laskemista. Summat ja keskiarvot ovat kyllä tärkeitä, mutta tärkein tieto saadaan tutkimalla tunnuslukuja ja trendejä eri ulottuvuuksien suhteen. Todellisen funktionaalisuuden saamiseksi sovelluksiin OLAP-järjestelmän on sisällettävä monipuolinen kieli erilaisten laskutoimitusten ja tilastollisten tunnuslukujen ilmaisemiseen [Tho97, Kos98].

2.2.3. Joustavuus

Järjestelmässä pitää olla joustavasti ilmaistavissa näkymät, laskentakaavojen määrittelyt, analyysit ja käyttöliittymät. Sen pitää myös tukea ennalta odottamattomia laskutoimituksia.

Näkymien joustavuus tarkoittaa sitä, että käyttäjä voi valita tiedon esitystavan vapaasti. Käyttäjän on myös voitava muuttaa numeroiden formaatteja, kaavojen määritelmiä ja tiedon lähdettä ilman kohtuutonta vaivaa tai haittaa. Käyttöliittymän yhteydessä joustavuus tarkoittaa samaa kuin intuitiivinen käyttöliittymä. Käytettävyyttä järjestelmälle saadaan hyvällä käyttöliittymällä, joka auttaa käyttäjää määrittelemään helposti monimutkaisiakin asioita. Nykypäivänä tämä aspekti on

yhä tärkeämpi, koska tietoa ei enää analysoi vain pieni joukko asialleen omistautuneita asiantuntijoita, vaan analysointityökalun pitää olla käyttökelpoinen myös varsinaisia tietojenkäsittelytekniikoita hallitsemattomalle loppukäyttäjälle [Tho97, Kos98].

2.3. TAULUKKOLASKENNAN JA SQL:N PUUTTEET

Tässä luvussa arvoidaan, miten taulukkolaskentaperustaiset järjestelmät ja SQL tukevat OLAP:lle luonteenomaista toiminnallisuutta. Luvussa keskitytään näkymän määrittelyn joustavuuden tarkasteluun.

2.3.1. Taulukkolaskenta ja OLAP

OLAP:n ilmaisuvoiman ja toiminnallisuuden toteuttaminen taulukkolaskentaperustaisilla järjestelmillä törmää tiedon esittämisen vaikeuteen. Luotaessa yksi suuri taulu ongelmaksi nousee yhteenvetotietojen laskemisen optimointi. Jos taas luodaan useampia pieniä tauluja, niin ongelmaksi nousee pirstaleisen tiedon tulkitsemisen vaikeus. Pienestäkin tietokannasta saadaan yksinkertaisilla kyselyillä satoja ellei tuhansia tauluja. Luotaessa yhteenvetotietoja näille saadaan aikaiseksi vielä lisää tauluja. Lisäksi joudutaan kopioimaan jokaiseen tauluun yhteenvetotietojen laskentakaavat. Yhteenvetotietoja luotaessa joudutaan lisäksi linkittämään jokainen yhteenvetotietoa laskettaessa käytetty tietoalkio erikseen asianmukaiseen yhteenvetotietoon.

Ongelmaksi nousee taulukkolaskentaperustaisten järjestelmien kyvyttömyys uudelleenorganisointiin yksinkertaisella tavalla. Uusi näkymä on luotava tyhjästä. Perimmäinen syy taulukkolaskentaperustaisten järjestelmien sopimattomuudelle on niiden kyvyttömyys erottaa rakenne näkymistä. Vaikka periaatteessa on mahdollista luoda hierarkkisia moniulotteisia näkymiä perinteisillä taulukkolaskentajärjestelmillä, on se käytännössä kuitenkin lähes mahdotonta [Tho97].

2.3.2. SQL ja OLAP

Tarkastellaan, onnistuuko SQL taulukkolaskentajärjestelmiä paremmin tukemaan OLAP:lle luonteenomaista toiminnallisuutta. SQL:n lähtökohtana on tarjota loppukäyttäjälle ilmaisuvoimainen ja intuitiivinen kieli.

SQL:n yhteenvetotaulujen ongelmana on niiden esitysmuoto, koska ne esitetään sarakkeina esitettyjen tekijoiden mukaan organisoituna. SQL ei myöskään pysty uudelleenjärjestämään tauluja eikä näkymiä. Olisi kyllä mahdollista koodata erikoistapaus –näkymä, jossa tieto esitettäisiin moniulotteisesti organisoituna, mutta se olisi hankalaa ja vaatisi uudelleen-koodausta arvojen muuttuessa. Samoin SQL ei tue joustavasti näkymien uudelleenorganisointia. Esimerkiksi edes budjetoidun ja todellisen myynnin suhde on mahdoton määritellä käymättä läpi joukkoa monimutkaisia operaatioita. SQL:ssä on myös muita rajoituksia. Yleisiä analyysifunktioita, kuten kumulatiivisten keskiarvojen ja kokonaismäärien, osasummien ja tiedon luokittelun ilmaisemista, ei tueta perus –SQL:ssä. Sen sijaan SQL3:een [Mel94] on kehitetty ilmaisuja, joilla simuloidaan moniulotteista kyselykieltä. Sen group-by rollup –ilmaisu luo valitun taulun pohjalta aggregointimuuttujia. Yksinkertaisessa tilanteessa tämä onnistuu hyvin, mutta intuitiivinen näkemys tietoon katoaa haluttaessa samaan tauluun eri perustein luotua aggregointitietoa. Tämänkaltainen näkymä pakottaa käyttäjän 'yksi-rivi-kerrallaan'-tyyppiseen tarkasteluun. Tarkasteltavaan esimerkkiin liittyvä taulu esitellään kuvassa 1.

K#	O#	MÄÄRÄ
K1	O1	300
K1	O2	200
K2	O1	300
K2	O2	400
K3	O2	200
K4	O2	200

Kuva 1: Esimerkkitaulu.

Taulussa ilmaistaan kauppojen (K1-K4) myymien osien (O1-O2) määrät (MÄÄRÄ). Taulun ensimmäinen rivi tulkitaan siten, että kaupassa K1 on myyty osaa O1 kolmensadan yksikön edestä. Kuvan taulu on tyypillinen SQL:n relaatiotaulu. Tämän taulun pohjalta voidaan luoda erilaisia summatauluja group-by rollup –ilmaisulla. Kuvassa 2 esitetään kaksi esimerkkitauluun liittyvää summataulua.

K#	KOK. MÄÄRÄ
K1	500
K2	700
K3	200
K4	200

O#	KOK. MÄÄRÄ
O1	600
O2	1000

Kuva 2: Erilaisia esimerkkitauluun (kuva 1) perustuvia summatauluja.

Kummassakin yhteenvetotaulussa myyntejä tarkastellaan yhden tekijän suhteen. Toisessa yhteenvetotaulussa niitä tarkastellaan kauppojen suhteen ja toisessa osien suhteen. Ensimmäisen taulun ensimmäinen rivi ilmaisee, että kaupassa K1 on myyty kaikkia osia yhteensä viidensadan yksikön edestä. SQL:n esitys ongelmat tulevat esiin, jos halutaan yhdistää group-by rollup –ilmaisulla samaan tauluun summia, joita tarkastellaan eri tekijöiden suhteen. Tällainen esimerkkiin liittyvä taulu annetaan kuvassa 3.

K#	O#	KOK. MÄÄRÄ
K1	null	500
K2	null	700
K3	null	200
K4	null	200
null	O1	600
null	O2	1000

Kuva 3: Yhteenvetotaulu, johon on yhdistetty kuvassa 2 olevat yhteenvetotaulut.

Taulussa (kuva 3) esiintyy nyt arvoja ’null’, joiden semanttinen merkitys on kuitenkin ’all’. Toisin sanoen ensimmäisellä rivillä esitetään kauppa K1:n kaikkien osien kokonaismyyntien summa. SQL3:n group-by cube –ilmaisulla annetaan lähtötaulun tietojen lisäksi kaikki mahdolliset yhteenvetotiedot, mitä tarkasteltavien tekijöiden välillä on mahdollista muodostaa. Kuva 4 ilmaisee group-by cube –ilmaisun tuottaman yhteenvetotaulun esimerkkitauluun liittyen.

K#	O#	KOK. MÄÄRÄ
K1	O1	300
K1	O2	200
K2	O1	300
K2	O2	400
K3	O2	200
K4	O2	200
K1	null	500
K2	null	700
K3	null	200
K4	null	200
null	O1	600
null	O2	1000
null	null	1600

Kuva 4: Group-by cube –ilmaisun tuottama yhteenvetotaulu kuvan 1 esimerkkitauluun perustuen.

SQL3:n group-by cube –ilmaisu on itseasiassa vain laajennettu group by rollup –ilmaisu. Group-by cube –ilmaisu johtaa kuitenkin samaan ongelmaan kuin group-by rollup –ilmaisu monimutkaisemmissa kyselyissä. Toisin sanoen taulu menettää intuitiivisen luettavuutensa. Perimmäinen syy SQL:n sopimattomuuteen moniulotteisten yhteenvetotietojen ilmaisemiseen on sen relationaalinen lähtökohta, joka ei pysty käsittelemään moniulotteisia tietokantoja tehokkaasti ja tarkoituksenmukaisella tavalla [Tho97]. SQL:n `CUBE`-operaattoria sekä sen optimointia on tutkittu julkaisussa [AAD+96]. Vastaavaa operaattoria on tarkasteltu julkaisussa [GBL+95]. Kuvassa 5 esitellään kuvan 4 SQL –taulu organisoituna kahden tekijän suhteen.

	O1	O2	KOK. MÄÄRÄ
K1	300	200	500
K2	300	400	700
K3	0	200	200
K4	0	200	200
KOK. MÄÄRÄ	600	1000	1600

Kuva 5: Kuvan 4 taulu esitettynä kahden tekijän (kauppojen ja osien) suhteen organisoituna.

SQL3:n lisäksi nD-SQL:ää [GL98] on ehdotettu moniulotteiseen yhteenvetotiedon organisointiin.

Taulukkolaskentaperustaiset järjestelmät ja SQL ovat siis osoittaneet kykenemättömyytensä organisoida yhteenvetotietoa luonnollisella tai lähes luonnollisella tavalla. Taulukkolaskentaperustaiset järjestelmät pystyvät tähän vain erittäin pienellä tietomäärällä. SQL:lla on mahdollista tehdä melko luonnollisella tavalla sellaisia yksinkertaisia kyselyjä, jotka perustuvat sarakkeisiin. Sen sijaan se

ei mahdollista analyysia tai rivien välisiä vertailuja. Kumpikaan lähestymistapa ei anna työkaluja monimutkaisten ja suurten tietomäärien käsittelemiseksi.

2.4. LÄHESTYMISTAVAT

OLAP-toiminnallisuus voidaan saada aikaan monella eri tavalla. Seuraavaksi tarkastellaan tiedon organisoinnin eroja. Perinteinen lähestymistapa ROLAP (Relational OLAP) perustuu relaatiomalliin. Tämän tavan erityispiirteenä on tiedon tallentaminen relaatiotauluihin (tables). Toinen perustapa organisoida OLAP on MOLAP (Multidimensional OLAP), jossa tieto talletetaan johonkin loogisesti moniulotteiseen tietorakenteeseen. Tälle tavalle tyypillistä on tiedon tallentaminen harvoihin tauluihin (sparse arrays). HOLAP (Hybrid OLAP) -lähestymistapa yhdistää ROLAP:n ja MOLAP:n tiedon tallentamistavat.

2.4.1. ROLAP (Relational OLAP)

ROLAP:ssa tieto talletetaan relaatiotietokannan tai laajennetun relaatiotietokannan tauluina. ROLAP –järjestelmät käyttävät SQL:ää, jolleivat aivan suoraan niin ainakin jokin yksinkertaisen konvertoijan kautta. MicroStrategyn Intelligence palvelin [Mic] sekä Informixin MetaCube [Inf a, Inf b] ovat esimerkkejä kaupallisista ROLAP -järjestelmistä. ROLAP –sovellusten hyvänä puolena on harvan tiedon pakkaamisen tehokkuus. Huonona puolena voidaan nähdä sisäiset, perustavaa laatua olevat ristiriidat SQL-tyyppisten kyselykielien sekä OLAP-toiminnallisuuden välillä [Tho97, ZDN97, Tam98, Dat00].

2.4.2. MOLAP (Multidimensional OLAP)

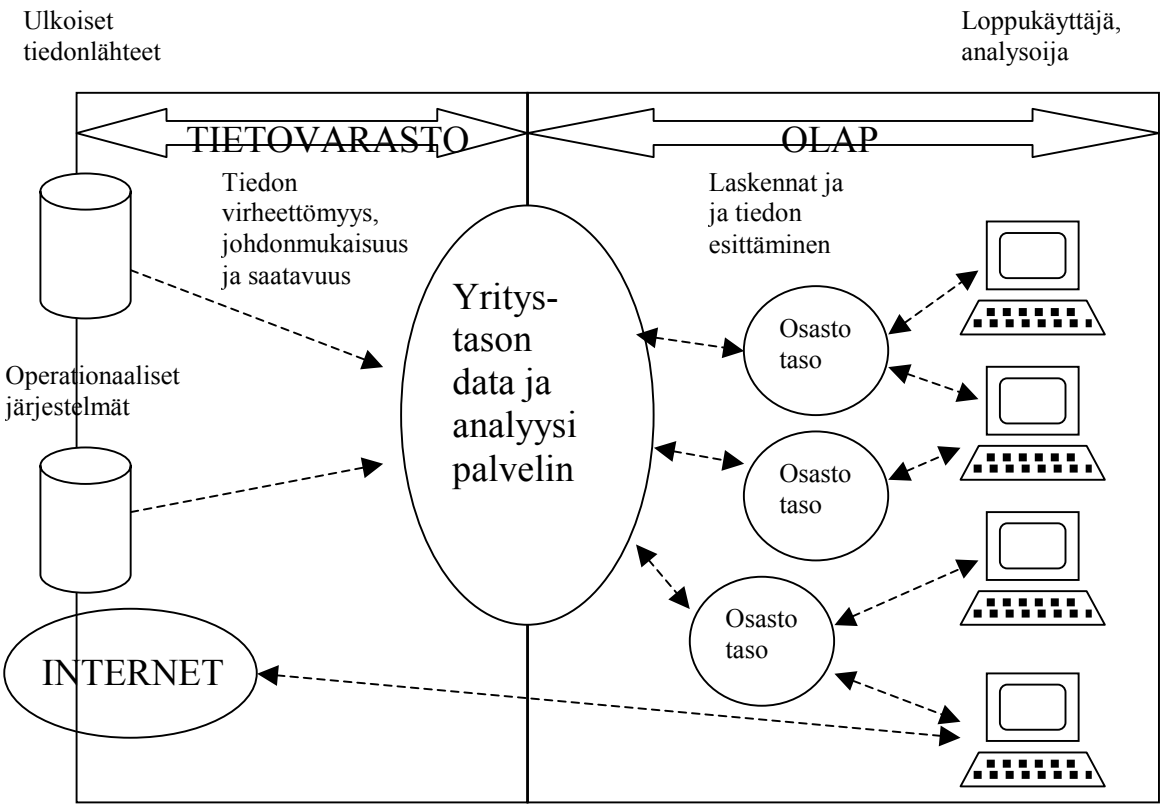
MOLAP tallettaa tiedon suoraan johonkin moniulotteiseen tietorakenteeseen, esim. harvaan tauluun. Hyperion Softwaren Essbase [Hyp] on eräs kaupallinen MOLAP-järjestelmä. Harvaan listaan talletetaan vain solujen arvot. Taulun indeksi kertoo arvon tarkan sijainnin moniulotteisessa avaruudessa. Solun etsiminen on yksinkertaista taulun läpikäymistä. MOLAP:ssa on selvästi tarvetta jollekin harvan tiedon pakkausmetodille. Siksi MOLAP on parhaimmillaan pienillä ja keskisuurilla tietokannoilla [Tho97, ZDN97, Tam98, Dat00]. Jotta MOLAP-järjestelmä toimisi tehokkaasti, on sekä sen indeksointiin että tilankäyttöön kiinnitettävä erityistä huomiota. Erilaisia indeksointitapoja (esim. [BKK96, GHRU99]) on MOLAP:n yhteydessä kehitetty runsaasti.

2.4.3. HOLAP (Hybrid OLAP)

HOLAP integroi ROLAP ja MOLAP-lähestymistavat. Siinä tyypillisesti harva taulu jaetaan harvoihin ja tiheisiin osiin. Harvat osuudet talletetaan ROLAP-lähestymistavalla, ja loppuihin luodaan MOLAP-tyyppinen indeksointi [TAM98, ZDN97]. HOLAP:ssa on tärkeää kehittää algoritmeja jotka, tunnistavat tiheitä osia tietokuutiosta. Eräs tällainen algoritmi on esitelty artikkelissa [AGG+98].

2.5. OLAP JA TIETOVARASTOT

Termit OLAP ja tietovarasto (data warehouse) määrittelevät joitakin osa-alueita ABDOP-kokonaisuudesta (Analysis-based decision-oriented processing). OLAP:n ja tietovarastojen perinteisissä määritelmissä on paljon päällekkäisyyttä. Näistä yhtäläisyyksistä huolimatta niissä on muutamia perustavaa laatua olevia eroavaisuuksia. Tietovarastot keskittyvät enemmänkin raakatiedon prosessointiin saadakseen sen oikeaksi, johdonmukaiseksi ja loppukäyttäjälle helpoksi hallita, kun taas OLAP keskittyy loppukäyttäjän analyyttisiin tarpeisiin ja raakatiedon mallintamiseen sekä analyysien ja yhteenvetotietojen laskemisprosesseihin. OLAP ei ota kantaa siihen, miten raakadata saadaan ja onko se oikeaa tai johdonmukaista. Tätä jakoa selvennetään kuvassa 6.



Kuva 6: OLAP:n ja tietovarastojen suhde [Tho97].

2.6. TUTKIELMAN ORGANISOINTI

Tutkielman loppuosa organisoidaan seuraavasti. Toisessa luvussa esitellään tutkielmassa käytettävä MOLAP –tietokuution käsitteistö ja organisointi. Kolmas luku käsittelee MOLAP –kuution toteuttamista logiikkaohjelmoinnin primitiiveillä. Neljännessä luvussa annetaan esimerkkinä käytettävä MOLAP –kuutio. Viidennessä luvussa esitellään MOLAP –kuution monipuoliseen käsittelyyn kehitetty view –operaatio. Kuudes luku tarkastelee MOLAP –taulun laajennukseen kehitettyä add –operaatiota. Kehitetty OLAP –kyselykieli koostuu yksinkertaisista logiikkaohjelmointi –ilmauksista, view- ja add –operaatioista sekä MOLAP –kuution ja relationaalisen tiedon integroinnissa käytettävästä predikaatista. Tähän kieleen perustuen luvussa 7 annetaan esimerkkikyselyjä. Kahdeksannessa luvussa esitellään kehitetty graafinen käyttöliittymä MOLAP –kyselyjen tekemiseen. Yhdeksännessä luvussa annetaan yhteenveto kyselykielestä.

3. MOLAP –tietokuution organisointi

OLAP:n uutuuden takia siihen liittyvä terminologia ei ole vielä vakiintunut. Tässä luvussa esitellään OLAP:n käsitteistä *muuttujat* ja *taulut* sekä niihin liittyvät käsitteet *solu* ja *kaava*. Muuttujat ovat tiedon esittämiseen ja organisointiin liittyviä käsitteitä, jotka sisältyvät elementteinä tauluihin. Taulu koostuu soluista, jotka ilmaisevat muuttujien arvoja. Taulun rivit ilmaisevat, miten eri muuttujien arvot liittyvät toisiinsa. Kaavat liittyvät niihin muuttujiin, joiden arvojen laskemiseksi käytetään matemaattisia kaavoja.

3.1. MUUTTUJAT

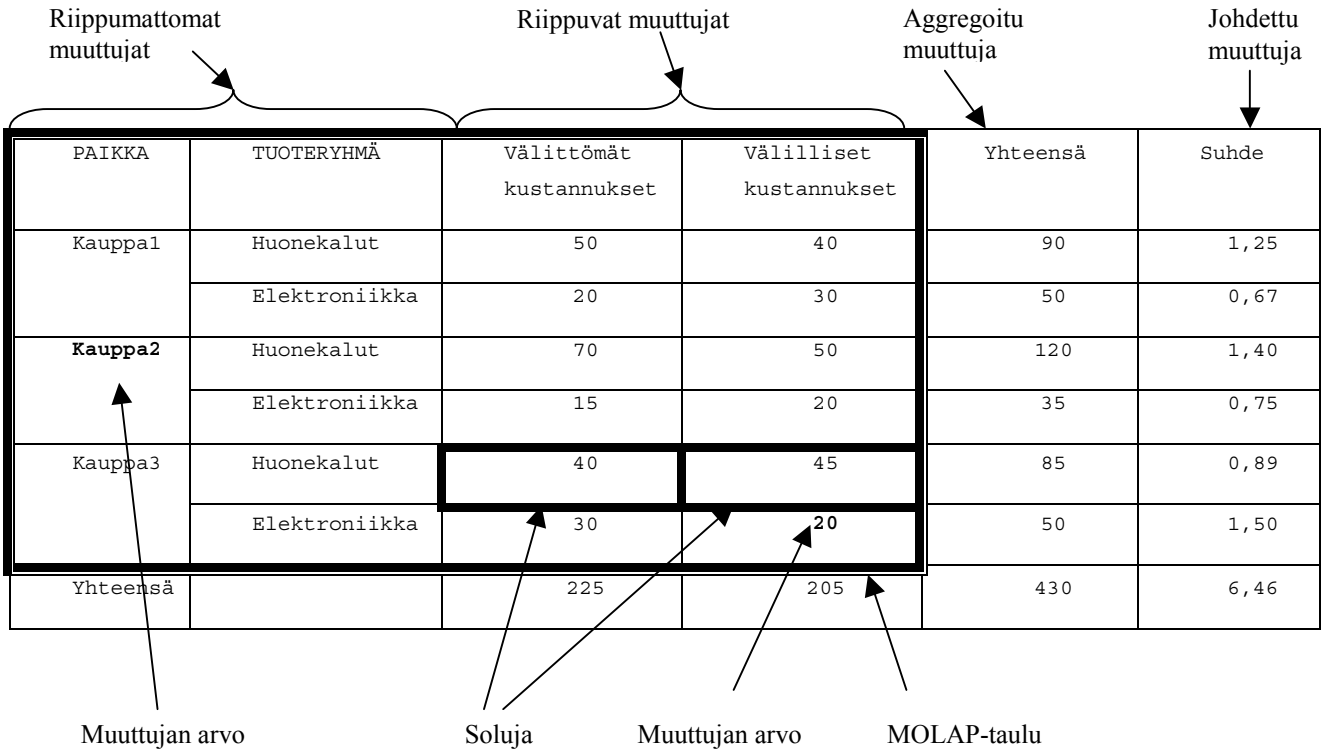
Date [Dat00] on esitellyt kaksi erilaista OLAP –muuttujan tyyppiä: *riippumaton muuttuja* ja *riippuva muuttuja*. Tässä yhteydessä käsite ’muuttuja’ tarkoittaa eri asiaa kuin ohjelmointikielten muuttuja –käsite. OLAP –muuttujaa käytetään viittaamaan joko taulussa olevaan yhteenvetotietoon tai ilmaisemaan sellainen tekijä, jonka mukaan yhteenvetotieto on organisoitu. Edellisiä muuttujia kutsutaan riippuviksi muuttujiksi ja jälkimmäisiä riippumattomiksi muuttujiksi. Lisäksi järjestelmään voi kuulua *aggregointimuuttujia* ja *johdettuja muuttujia*.

Aggregointimuuttujalla ymmärretään riippuvien muuttujien arvojen sarake- ja rivisummia. Aggregointimuuttujalla ilmaistaan myös kaikkien riippuvien muuttujien arvojen yhteissumma. Täten se ilmaisee taulun sisältämien yhteenvetotietojen kokonaissumman. Johdetuille muuttujille, toisin kuin aggregointimuuttujille, on tyypillistä, että niissä voidaan käyttää myös taulun ulkopuolista informaatiota.

Riipumattomia muuttujia kutsutaan yleisesti myös sovelluksen *ulottuvuuksiksi* (dimension) tai *ulottuvuusmuuttujiksi* (dimension variable) ja riippumattomia muuttujia *ei –ulottuvuusmuuttujiksi* (non dimensional variable) [Tho97]. Riippuvien muuttujien arvot taulussa liittyvät aina tiettyyn riippumattomien muuttujien arvojen kombinaatioon. MOLAP –taululle on tyypillistä, että se voi sisältää useampia riippumattomia muuttujia ja riippuvia muuttujia yhtäaikaa. Toisin sanoen, riippumattomat muuttujat yhdessä määrittelevät kaikki ne tekijät, joiden perusteella taulun soluissa oleva yhteenvetotieto on organisoitu. Soluissa esitetään

yhteenvetotietoa kuvaavien muuttujien eli riippuvien muuttujien arvoja. MOLAP –taulussa kuvataan riippuvat muuttujat, riippumattomat muuttujat ja solut.

Muuttujat voivat olla tyypeiltään erilaisia. Tyyppejä ovat esimerkiksi *nominaaliset-*, *ordinaaliset-* ja *kardinaaliset* muuttujat. Nominaaliselle muuttujalle on tunnusomaista, että sen arvoja ei voida laittaa järjestykseen. Esimerkkinä nominaalisesta muuttujasta on väri. Tämän tyyppisille muuttujille voidaan käytännössä laskea vain kokonaismäärä. Sen sijaan ordinaalisen muuttujan arvoilla on järjestys, mutta niiden arvojen välillä ei voida suorittaa laskutoimituksia. Esimerkki ordinaalisesta muuttujasta on sijaluku. Kardinaalisen muuttujan arvot voidaan järjestää keskenään. Sen lisäksi arvojen kesken voidaan suorittaa laskutoimituksia. Esimerkkeinä kardinaalisesta muuttujasta ovat paino kiloina ja aika sekunteina ilmaistuna.



PAIKKA	TUOTERYHMÄ	Välittömät kustannukset	Välilliset kustannukset	Yhteensä	Suhde
Kauppa1	Huonekalut	50	40	90	1,25
	Elektroniikka	20	30	50	0,67
Kauppa2	Huonekalut	70	50	120	1,40
	Elektroniikka	15	20	35	0,75
Kauppa3	Huonekalut	40	45	85	0,89
	Elektroniikka	30	**20**	50	1,50
Yhteensä		225	205	430	6,46

Kuva 7: Kokonaiskustannuksiin liittyvä MOLAP-taulu.

Esimerkissä välittömiä ja välillisiä kokonaiskustannuksia tarkastellaan kauppojen ja tuoteryhmien perusteella. Kuvassa 7 on esitetty esimerkkiin liittyvä MOLAP –taulu. Esimerkkitaulussa riippuvina muuttujina ovat `Välittömät` ja `Välilliset kustannukset`. Ne ovat organisoitu riippumattomien muuttujien `PAIKKA` ja `TUOTERYHMÄ` suhteen. Riippumattomalla muuttujalla `PAIKKA` on arvot `Kauppa1`,

Kauppa2 ja Kauppa3. Kun taas riippumattoman muuttujan TUOTERYHMÄ arvot ovat Huonekalut ja Elektroniikka. Riippuvien muuttujien arvot ovat kokonaislukuja ja ne kuvaavat kokonaiskustannuksia kaupoittain ja tuoteryhmittäin esitettynä. Ensimmäisellä rivillä ilmaistaan, että Kauppa1:ssä on tuoteryhmästä Huonekalut koituneet Välittömät kustannukset 50 yksikköä, ja Välilliset kustannuket 40 yksikköä.

Esimerkissä aggregointimuuttujalla Yhteensä ilmaistaan sekä sarake- että rivisummat. Johdettu muuttuja Suhde ilmaisee riippumattomien muuttujien Välittömät kustannukset ja Välilliset kustannukset suhteen. Ensimmäisellä rivillä aggregoidun muuttujan arvo 90 ilmaisee, että Kauppa1:ssä on tuoteryhmästä Huonekalut koitunut kustannuksia *yhteensä* 90 yksikön edestä. Viimeisenä ensimmäisellä rivillä on johdetun muuttujan Suhde arvo 1,25, joka on jakolaskun 50/40 lopputulos. Toisin sanoen Kauppa1:ssä tutoteryhmästä Huonekalut koituvat Välittömät kustannukset ovat 25% suuremmat kuin Välilliset kustannukset.

3.2. TIEDON TYYPIT

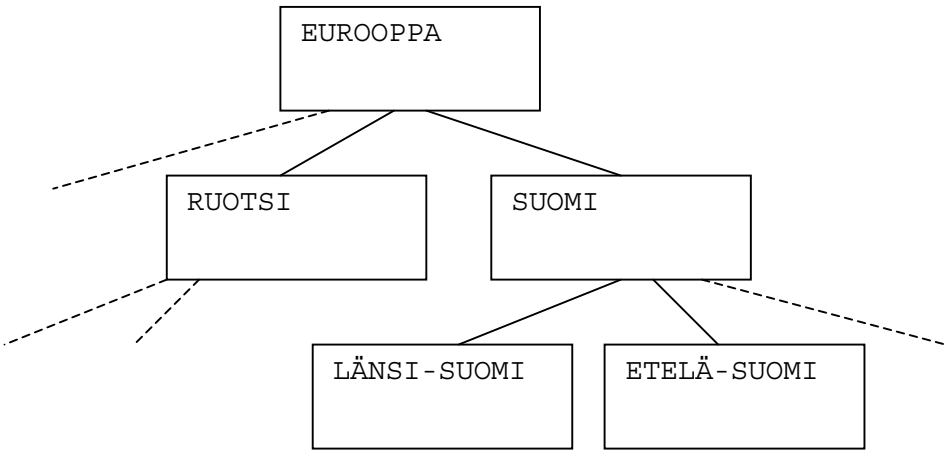
Vaikka riippuvan muuttujan arvot esitellään esimerkissä kokonaislukuina, voivat niiden arvot olla minkä tyyppistä informaatiota tahansa. Esimerkikisi arvoina voi olla ääntä tilanteessa, jossa riippuvana muuttujana on linnunlaulu, ja riippumattomina muuttujina laji, alue ja aika. Numeerinen informaatio on kuitenkin yleisintä, koska se takaa OLAP -järjestelmän aggregointimuuttujiin liittyvän ilmeisen tulkinnan [Tho97]. Esimerkiksi edellä mainitussa tapauksessa aggregointimuuttuja ei olisi ilmeisellä tavalla tulkittavissa.

3.3. TAULUT

MOLAP –kuutio on MOLAP –taulujen, ominaisuustaulujen ja aputietotaulujen kokoelma. MOLAP –tietokuutiossa MOLAP –tauluja ja riippumattomien muuttujien ominaisuuksia esittäviä ominaisuustauluja kutsutaan *kantatauluiksi.* Muita taulutyyppejä ovat *aputietotaulut* ja *virtuaalitaulut*. Kantataulut ovat MOLAP –tietokuution perustauluja ja ne sisältävät yhteenvetotietoa. Loppukäyttäjän kyselyt kohdistuvat kantatauluihin. Kantatauluja ovat siis ne taulut, jotka luodaan tietokannassa/tietokannoissa olevien raakatietojen pohjalta. Näitä tauluja päivitetään

tietyin väliajoin, eikä käyttäjällä ole mitään tapaa muuttaa kantatauluja. Kantataulujen päivittäminen on tämän työn ulkopuolella, ja työssä oletetaan, että käytettävissä on aina jokin kantataulujen kokoelma.

MOLAP –taulut ovat moniulotteisia tauluja, jotka organisoivat riippumattomien muuttujien avulla riippuvien muuttujien sisältämän yhteenvetotiedon. Kyselyt kohdistuvat pääasiassa MOLAP -tauluhin. Mielekkään MOLAP –taulun riippumattomien ja riippuvien muuttujien määrän määrittelee kohdealue, jonka kuvaamiseen taulu on suunniteltu.



Kuva 8: Esimerkki karkeistushierarkiasta.

Aputietotaulut määrittelevät tietojärjestelmän tarvitsemaa tai muuten merkityksellistä tietoa, joka ei ole määriteltävissä kantataulujen avulla. Keskeisin aputietotaulujen tyyppi on *karkeistushierarkian esitystaulu*, joka määrittelee *karkeistushierarkiat* riippumattomille muuttujille. Esimerkki karkeistushierarkiasta on esitetty kuvassa 8.

Karkeistushierarkioiden määritteleminen riippumattomille muuttujille on erittäin tärkeä ja analysointia helpottava toimenpide. Se mahdollistaa yhteenvetotiedon tarkastelemisen eri kokonaisuuksiin perustuen eli siihen perustuen tietojen esitystapoja voidaan sekä karkeistaa että tarkentaa. Lisäksi se helpottaa tiedon organisointia ja yhteenvetotietojen laskemista.

Karkeistamishierarkiassa esitetään hierarkioiden rakenteet MOLAP –kuutiossa. Sen avulla voidaan yksikäsitteisesti määritellä riippumattoman muuttujan arvojen suhde sisaruksiinsa ja hierarkiassa sen ylä- ja alapuolella oleviin arvoihin.

Virtuaalitaulut ovat niitä tauluja, jotka syntyvät operaatioiden toteuttamisen tuloksena kantatauluista. Virtuaalitaulujen päätyyppi on *näkemystaulu*, joka syntyy kyselyn tuloksena. Näkemystaulut voivat olla myös lähtökohtana uusille lisäkyselyille tiedon tarkempaa analysointia varten. Tällöin niiden perusteella luodaan uusia näkemystauluja. Näkemystaulut ovat rakenteeltaan samankaltaisia kuin MOLAP –taulut.

Riippumattomiin muuttujiin liittyy monesti myös sellaista tietoa, mikä pitää eksplisiittisesti ilmaista, mutta joka ei ole suoraan liitettävissä tauluun riippumattomana muuttujana. Tämän vuoksi luodaan jokaiselle riippumattomalle muuttujalle oma ominaisuustaulu, johon talletetaan ko. muuttujaan liittyviä ominaisuuksia eli attribuutteja. Attribuutin arvo riippuu yhden riippumattoman muuttujan arvosta. Esimerkiksi `myyjä` –ulottuvuuteen voidaan liittää attributit `nimi`, `osoite` ja `palkka`. Ominaisuustaulut esitetään kuten relaatiot relaatiotietokannassa.

3.4. MUUTTUJIEN PÄIVITYKSESTÄ

Muuttujien päivitykset ovat harvinaisia tilanteita, mutta joskus tällaisiakin tilanteita esiintyy. Päivitykset muuttujien joukossa saattavat johtua OLAP –ympäristön tuottamasta uudesta informaatiosta, jonka avulla voidaan luokitella esimerkiksi `kokonaismyynti` myös ostajaryhmittäin. Tässä tilanteessa taulu, jossa myös myynnit esitetään, joudutaan päivittämään uuden ulottuvuuden lisäämiseksi. Tällöin on tarkasteltava myös mahdollisia muita samassa taulussa määriteltyjä riippuvia muuttujia.

Riippumattoman muuttujan päivitykset tulevat kyseeseen silloin, kun sen arvojen lukumäärä muuttuu. Jos esimerkissämme kauppaketju laajenee uudella kaupalla, on tauluun lisättävä riippumattomalle muuttujalle `paikka` uusi arvo.

Attribuuttien päivitykset saattavat olla ongelmallisia tilanteita. Päivitys saattaa johtaa tilanteeseen, jossa esimerkiksi johdettujen muuttujien laskeminen ei ole enää triviaali prosessi. Esimerkiksi jos `kauppa2` laajentaa tilojaan. Tämän seurauksena siihen liittyvä attribuutti `pinta-ala` joudutaan päivittämään. Tästä seuraa tilanne, jossa esimerkiksi johdetun muuttujan arvo, joka lasketaan jakamalla `pinta-ala` myyntien määrällä, ei ole yksinkertaisesti laskettavissa. Jotta tätä johdettua muuttujaa voidaan tutkia riippumatonta muuttujaa `aika` vasten on myös vanha

`pinta-ala` oltava tallessa, jotta päivitystä edeltävät johdetun muuttujan arvot olisivat oikein. Päivitykset attribuuteissa voivat siis vaikuttaa myös muihin muuttujien arvoihin. Yksi mahdollisuus tilanteen korjaamiseksi on luoda `paikka` – muuttujalle arvo `vanha_kauppa2`. Tämä arvo kuvaa `kauppa2`:n tilaa ennen päivitystä ja `paikka` –muuttujan arvo `kauppa2` kuvaa `kauppa2`:ta päivityksen jälkeen. Toinen, keinotekoisempi, ratkaisu on muuttaa `pinta-ala` riippumattomaksi muuttujaksi, joka määräytyy `paikan` ja `ajan` mukaan.

3.5. SOLUT

Taulussa olevien riippumattomien muuttujien erilaisten arvojen yhdistelmä määrittelee yksiselitteisesti yhden *soluksi* kutsutun tietokentän, joka voi olla yksi- tai monipaikkainen. Tämän tietokentän paikkoihin talletetaan riippuvien muuttujien arvoja. Todellisissa tietokannoissa osa näistä soluista on tyhjiä joko siksi, että kyseisten riippumattomien muuttujien arvojen rajaama tapaus ei ole *mielekäs* jollakin sovellusalueella, tai siksi, että tieto *puuttuu* (missing value).

Esimerkiksi solu jossa arvona on myyjä1:n myymien osa2:n määrä, ja myyjä1 työskentelee liikkeessä ei ole myynnissä osa2:ta. Tämä solu ei ole mielekäs, koska myyjä1:n ei ole mahdollista myydä osa2:ta.

Tiedon puuttuminen voi olla seurausta esimerkiksi muuttujien arvoihin liittyvistä erilaisista tiedon tallennuskäytännöistä tai sellaisesta virheestä solussa olevassa tiedossa, joka ei ole normaalisti mahdollista arvolle. Tallennuskäytännöistä johtuvat puuttuvat arvot liittyvät karkeistushierarkioihin. Esimerkiksi jotkut liikkeet voivat laskea myynnit tunneittain, kun joissakin liikkeissä myynnit lasketaan päivittäin. Tästä seuraa tilanne, jossa toisen kaupan myyntejä voidaan tarkastella vain päivätasolla, kun taas ensimmäisessä kaupassa myyntejä voidaan analysoida tuntitasolla. Tämä tarkoittaa sitä, että karkeistushierarkiassa päivätasoa yksityiskohtaisemmat arvot puuttuvat joidenkin liikkeiden osalta.

			PAIKKA					
			kauppa 1		kauppa 2		kauppa 3	
AIKA			todellinen	budjetoitu	todellinen	budjetoitu	todellinen	budjetoitu
vuosi 1999	1.puolikas	1. Neljännes	1371	1290	1531	1750		
		2. Neljännes	1237	1290		1750	1678	1790
	2.puolikas	3. Neljännes			1681	1750	1768	1790
		4. Neljännes	1391	1290			2342	1790

Kuva 9: Esimerkki harvasta tietokannasta.

Tietokantaa jossa esiintyy tarkoituksettomia tai puuttuvia arvoja kutsutaan *harvaksi* (sparse) tietokannaksi. Esimerkkinäkymä harvasta tietokannasta annetaan kuvassa 9. Harvoissa tietokannoissa ongelmana on tarkoituksettoman tai puuttuvan arvon tulkinta. Puuttuva arvo voi olla mahdollisesti tulossa, esimerkiksi myöhästynyt myyntiraportti. Edelleen kyseessä voi olla tilanne, jossa soluun ei ole mahdollista tulla arvoa, kuten esimerkiksi jos `kauppa3` on avattu vasta `2. neljänneksellä` (ks. kuva 9). Puuttuva arvo voi tarkoittaa myös samaa kuin nolla siinä tapauksessa, että kyseistä tuotetta ei ole myyty kyseisenä aikana.

3.6. KAAVAT

Kaavoja käytetään laskemaan johdettuja muuttujia. Tässä työssä on otettu periaatteeksi, että tiedon määrä pidetään minimaalisena. Tämä tarkoittaa sitä, että kaikki johdetut muuttujat lasketaan vasta silloin, kun niitä tarvitaan. Minimaalista metodia kutsutaan myös *laiskaksi* (lazy) metodiksi. Joissakin lähestymistavoissa lasketaan kaikki johdettujen muuttujien arvot etukäteen. Tämä pienentää vasteaikoja ja on käytännöllistä suurten tietomassojen kyseessä ollessa. Tätä lähestymistapaa kutsutaan *innokkaaksi* (eager) metodiksi. Innokkaassa metodissa ongelmaksi nousee hidastuvat päivitykset, koska päivityksen yhteydessä myös johdetut muuttujat pitää laskea uudestaan.

Laiskan- ja innokkaan metodin yhdistelmät nk. *puoli –innokkaat* (semi –eager) metodit laskevat jotkut johdetuista muuttujista valmiiksi. Loput johdetut muuttujat, joita käytetään todennäköisesti vähemmän, lasketaan vain tarvittaessa.

Moniulotteisessa ympäristössä kaavat toimivat abstraktimmalla tasolla kuin perinteisessä taulukkolaskentaympäristössä. Sen sijaan että ne määriteltäisiin yksittäisille soluille, ne määritellään riippumattomien muuttujien arvoihin liittyen. Tämä tarkoittaa, että kaavat pätevät kaikkiin soluihin, jotka jakavat kyseisten riippumattomien muuttujien arvot.

.

4. OLAP-kuutioiden esittäminen konstruktori –orientoituneesti logiikkaohjelmoinnissa

Monet tutkijat ovat ehdottaneet logiikkaohjelmointipohjaista lähestymistapaa tietokantojen ja tietojärjestelmien toteuttamiseen [Ull88, Liu99, SS94]. Tässä tutkielmassa kehitetään loppukäyttäjälle OLAP -analysointia helpottavia operaatioita, jotka voidaan liittää osaksi normaalin logiikkaohjelmoinnin avulla esitettyjä kyselyjä. Näiden operaatioiden avulla loppukäyttäjän kyselyjen tekeminen helpottuu, kun kaikki mutkikkaat (esim. rekursiiviset) määrittelyt liittyvät näiden operaatioiden toteuttamiseen ja käyttäjän ei tarvitse hallita niiden määrittelyä. On riittävää, että hän kykenee soveltamaan niitä.

4.1. KONSTRUKTORIT

Konstruktorien käyttöön tietojen rakenteellisessa mallintamisessa on kiinnitetty huomiota oliotietokantojen, deduktiivisten tietokantojen ja deduktiivisten oliotietokantojen yhteydessä [Ull88, Liu99]. Oliotietokantojen ja OLAP:n yhteenliittämistä on tutkittu esimerkiksi julkaisussa [GPS00].

Tiedon esittämiseksi OLAP –sovelluksissa järjestelmällisellä tavalla käytetään konstruktoreita, joilla on tietty muoto ja merkitys. Logiikkaohjelmoinnissa on mahdollista käyttää loogisia termejä esittämään *järjestetty joukko*, *järjestämätön joukko* ja *atomi*. Joukot esitetään tässä työssä rekursiivisesti, eli joukkojen alkiona voi olla edelleen uusia joukkoja. Joukkojen avulla voidaan siis määritellä monimutkaista sisäkkäistä tietoa. Tieto esitetään atomeina.

4.1.1. Järjestetty joukko

Järjestetty joukko on joukko alkiota joiden keskinäinen järjestys on oleellinen. Logiikkaohjelmoinnissa se voidaan esittää yhdistettynä terminä functor(A_1, A_2,..A_n), missä A_1, A_2, … , A_n ovat joukon alkioita ja 1, 2, … , n ilmaisee alkioiden järjestyksen joukossa. Functor on yhdistetyn termin nimi. Joukon alkiona voi olla atomeita, toisia järjestettyjä joukkoja tai järjestämättömiä joukkoja.

4.1.2. Järjestämätön joukko

Joukko, jonka alkioiden keskinäisellä järjestyksellä ei ole merkitystä, voidaan kuvata listana [A_1, A_2, … , A_n], missä A_1, A_2, … , A_n ovat alkiota. Joukkoa siis

emuloidaan logiikkaohjelmoinnin listana. Joukon alkiona voi olla atomeita, järjestettyjä joukkoja tai muita järjestämättömiä joukkoja. Tässä yhteydessä esitystapaa ajatellaan joukkona listan sijasta. Tämä tarkoittaa sitä, että jotkut listalle määritellyt primitiivit voidaan tulkita joukko –opillisiin käsittein. Esimerkiksi primitiivi member(X, Y), joka tarkistaa onko alkio X listan Y jäsen, voidaan tulkita joukko-opillisena ilmauksena siten, että X on joukon Y alkio. [NJ91].

4.2. RAKENNETTAVAN MOLAP-KUUTION PERUSLÄHTÖKOHDAT

Tietojärjestelmässä päätöksentekoa avustamaan kehitetään MOLAP –kuutio. Tässä työssä on tarkoitus lisäksi määritellä joukko loppukäyttäjää helpottavia operaatioita tietokuution analysoimiseksi. Jokainen operaatio saa operandikseen kuution ja tuottaa uuden kuution. Koska tämän tutkimuksen operaatiot ovat vapaasti yhdisteltävissä ja liitettävissä toisiinsa, vältytään monien olemassaolevien tuotteiden ‘yksi-operaatio-kerrallaan’ lähestymistavalta.

Tutkimuksessa kehitettävät operaatiot ovat minimaalisia, eli mitään operaatiota ei voida esittää muiden operaatioden avulla. Lisäksi mitään operaatiota ei voi poistaa menettämättä välttämätöntä analysointikykyä. Kehitettävä malli käsittelee symmetrisesti sekä riippumattomia että riippuvia muuttujia. Kehitettävät tekstuaalinen ja graafinen käyttöliittymä pidetään loogisesti erillään toisistaan. Toisin sanoen graafinen käyttöliittymä ei ole riippuvainen tekstuaalisesta käyttöliittymästä, eikä tekstuaalinen käyttöliittymä ole riippuvainen graafisesta käyttöliittymästä. Samankaltaiset periaatteet on todettu hyviksi myös lähteessä [AGS97].

4.3. OLAP –KUUTION LOGIIKKAPOHJAINEN ESITTÄMINEN

Seuraavaksi tarkastellaan työn konstruktiivista toteuttamista. Prototyyppi on toteutettu LPA:n Win-Prolog:lla. Tutkimuksen keskeisenä tavoitteena on kehittää logiikkaohjelmointityyppinen kieli loppukäyttäjälle OLAP –kyselyjen tekemiseen. Jatkossa oletetaan, että lukija on jonkin verran perehtynyt Prologin syntaksiin ja on täten tutustunut esimerkiksi jaetun muuttujan käsitteeseen (ks. esim. [SS94]).

Taulut esitetään järjestettyjen joukkojen kokoelmana. Operaatioden yleinen määrittely edellyttää sitä, että tauluista esitetään myös metatietoa, joka määrittelee taulun järjestettyjen joukkojen muodon ja argumenttien semanttisen merkityksen

eksplisiittisesti. Tätä metatietoa kutsutaan taulun *kaaviotasoksi* (schema level) ja järjestettyjen joukkojen kokoelmaa kutsutaan *ilmentymätasoksi* (instance level).

4.3.1. MOLAP –taulujen esittäminen logiikkaohjelmointiperustaisesti

Jokaisen MOLAP –taulun kaaviotaso esitetään seuraavana järjestettynä joukkona: table_descr(N, R, M), missä

- N (eräs atomi) ilmaisee sen MOLAP –taulun nimen, jonka nimisistä faktoista ilmentymätaso koostuu.
- R on järjestämätön joukko [dim(u_1, n_1), dim(u_2, n_2), … , dim(u_m, n_m)], joka liittää kyseisessä taulussa olevaan jokaiseen riippumattomaan muuttujaan u_i ($1 \leq i \leq m$) jonkin kokonaisluvun n_i. Tätä informaatiota tarvitaan (tarkastellaan myöhemmin), kun riippumattomiin muuttujiin liittyviä arvoja käsitellään ilmentymätasolla.
- M on järjestämätön joukko [dep(r_1, n_{m+1}), dep(r_2, n_{m+2}), … , (dep(r_k, n_{m+k})], joka liittää kyseessä olevan MOLAP-taulun jokaiseen riippuvaan muuttujaan r_i, ($1 \leq i \leq k$), kokonaisluvun n_{m+i}. Tätäkin informaatiota tarvitaan (käsitellään myöhemmin) riippumattomiin muuttujiin liittyvien arvojen käsitellään ilmentymätason käsittelemiseksi.

  MOLAP-tauluun, jonka nimi on xyz (yo. `table_descr`:n 1. argumentti), liittyvä ilmentymä esitetään kokoelmana muotoa xyz(v_1, v_2, … , v_{m+k}) olevia faktoja (järjestettyjä joukkoja). Niissä v_1, … ,v_{m+k} ovat muuttujiin liittyviä arvoja siten, että v_i ($1 \leq i \leq m+k$) on sen muuttujan arvo, johon liittyy nimi n_i kaaviotasolla. Tämä on se mekanismi, jolla muuttujat ja niiden arvot, eli kaavio- ja ilmentymätasot sidotaan toisiinsa.

```
table_descr(kustannukset, [dim(paikka, 1), dim(tuote, 2)],
[dep(välittömät_kust, 3), dep(välilliset_kust, 4)]).
kustannukset(kauppa1, elektroniikka, 20, 30).
kustannukset(kauppa1, huonekalut, 50, 40).
kustannukset(kauppa2, elektroniikka, 15, 20).
kustannukset(kauppa2, huonekalut, 70, 50).
kustannukset(kauppa3, elektroniikka, 30, 20).
kustannukset(kauppa3, huonekalut, 40, 45),
```

Kuva 10: MOLAP-taulun esittäminen Prologissa.

Kuvassa 10 annetaan esimerkkinä eräs MOLAP –taulu nimeltä `kustannukset`, jonka kaaviotaso ylläolevan perusteella esitetään `table_descr` –nimisenä faktana. Tämä taulu on sama taulu kuin kuvassa 7. Kuten jo aikaisemmin mainittiin, periaatteena tässä työssä on pitää tiedon määrä minimaalisena. Eli mitään tietoa, mikä voidaan laskea ei esitetä tauluissa. Tämän vuoksi kuvassa 10 esitetyssä taulussa ei ole kuvassa 7 esitettyjä johdettuja ja aggregoituja muuttujia. Taulussa on kaksi riippumatonta muuttujaa, `paikka` ja `aika`. Näiden muuttujien arvot esitetään ilmentymätason `kustannukset` –faktoissa 1. ja 2. argumenttina. Kaaviotason kolmannessa argumentissa esitellään kaksi riippuvaa muuttujaa, `välittömät kustannukset` ja `välilliset kustannukset`. Ilmentymätasolla näiden muuttujien arvot ovat 3. ja 4. argumenttina. Faktoissa kolmannen ja neljännen argumentin arvot siis määräytyvät ensimmäisen ja toisen argumentin arvojen perusteella.

4.3.2. Ominaisuustaulut

Ominaisuustauluilla ilmaistaan tietoa riippumattomiin muuttujiin liittyvistä ominaisuuksista. Prototyypissä ominaisuustaulut esitetään kuten relaatiot relaatiotietokannassa. Näiden taulujen käsittelyssä ei tarvita voimakkasta analysointia. Niihin kohdistuva kyselytarve voidaan tyydyttää perinteisellä relaatiotietokantojen ilmaisuvoimalla.

Mikä tahansa ominaisuustaulu esitetään relaationa, jolle annetaan sekä ilmentymä- että kaaviotason kuvaus. Ominaisuustaulun kaavio esitetään järjestettynä joukkona relation_descr(N, D, R), missä

– N on ominaisuustaulun nimi.

– D on yksialkioinen joukko [dim(Dim, 1)], jossa Dim on riippumattoman muuttujan nimi, johon ominaisuudet liittyvät. Numero 1 ilmaisee muuttujan arvon sijainnin ilmentymätasolla.

– R on järjestämätön joukko [rel($attr_1$, 2), rel($attr_2$, 3), … , rel($attr_k$, k+1)], joka liittää jonkin kokonaisluvun i ($2 \leq i \leq k+1$) attribuutteihin $attr_1$, $attr_2$, … , $attr_k$. Tämä kokonaisluku ilmaisee yksiselitteisesti attribuutin arvon paikan ilmentymätasolla.

Ominaisuustaulun ilmentymätaso esitetään joukkona faktoja (järjestettyjä joukkoja), joiden nimenä käytetään taulun kaavion yhteydessä annettua nimeä. Ensimmäisenä argumenttina on riippumattoman muuttujan arvo ja sitä seuraavat argumentit ovat kyseiseen arvoon liittyviä attribuuttien arvoja. Nämä arvot on järjestetty taulun kaaviotasolla ilmaistussa järjestyksessä kuten MOLAP –tauluissakin.

```
relation_descr(myyjien_tiedot, [dim(myyja, 1)],
[rel(myyjan_nimi, 2), rel(myyjan_osoite, 3), rel(palkka, 4)]).
myyjien_tiedot(yksi, arttu, kotikatu, 15000).
myyjien_tiedot(kaksi, liisa, puistokatu, 15000).
myyjien_tiedot(kolme, leena, rantatie, 9000).
myyjien_tiedot(nelja, mauno, keskuskatu, 10000).
```

Kuva 11: Ominaisuustaulun logiikkapohjainen esittäminen.

Kuvassa 11 annetaan esimerkkinä ominaisuustaulun `myyjien_tiedot` esitystapa. Kaaviotason kuvauksessa ensimmäisessä argumentissa ilmaistaan ominaisuustaulun nimi eli `myyjien_tiedot`. Seuraavassa argumentissa annetaan sen riippumattoman muuttujan nimi, `myyja`, mihin taulu liittyy. Kolmannessa argumentissa luetellaan attribuutit `myyjan_nimi`, `myyjan_osoite` ja `palkka`, jotka esittävät riippumattomaan muuttujaan `myyja` liittyvät ominaisuudet.

Ominaisuustaulun ilmentymätaso koostuu tässä tapauksessa `myyjien_tiedot` nimisten faktojen kokoelmasta. Näiden faktojen ensimmäisenä argumenttina on kaaviotasolla esitellyn riippumattoman muuttujan arvo, johon seuraavissa argumentteissa annetut attribuuttien arvot liittyvät. Esimerkiksi myyjän yksi attribuuteilla `myyjan_nimi`, `myyjan_osoite` ja `palkka` ovat arvot `arttu`, `kotikatu` ja `15000`.

4.3.3. Aputietotaulut

Karkeistamistaulut esittävät riippumattomaan muuttujaan liittyvän hierarkian. Nämä taulut ovat merkityksellisiä, koska tämänkaltaiset hierarkiat ovat välttämättömiä, jotta voitaisiin luoda eri yksityiskohtaisuuden tasoilla olevia yhteenvetotietoja luontevasti.

Yleisesti ottaen karkeistuksen kohde C jaetaan kaaviotasolla alikarkeistuksiin. Tämä esitetään järjestettyjen joukkojen kokoelmana, joilla on muoto granularity_schema(D, C, C'). Siinä D on sen riippumattoman muuttujan nimi, johon karkeistushierarkia liittyy. Ja C' on C:n välitön alikarkeistus. Jokainen karkeistustaso esitetään samalla tavalla. Karkeistamishierarkian alin taso saavutetaan silloin, kun jollekin karkeistuksen kohteelle ei ole kuvattu välitöntä alikarkeistusta.

Karkeistamistaulun ilmentymätaso koostuu muotoa granularity_instance(L_1, L_2) olevista faktoista (järjestetyistä joukoista). L_1 ja L_2 kuvaavat kaaviotasolla esitettyjen karkeistustasojen ilmentymiä. Ilmaisu tulkitaan siten, että karkeistustason ilmentymä L_2 sisältyy välittömästi karkeistustason ilmentymään L_1.

```
granularity_schema(paikka, maa, alue).
granularity_schema(paikka, alue, kauppa).
granularity_instance(suomi, etelä).
granularity_instance(suomi, itä).
granularity_instance(itä, kauppa1).
granularity_instance(etelä, kauppa2).
granularity_instance(etelä, kauppa3).
```

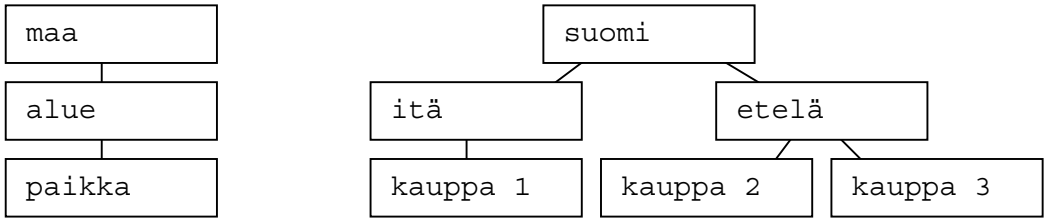
Kuva 12: Karkeistamistaulun logiikkapohjainen esittämistapa.

Kuvassa 12 on esitelty esimerkkinä riippumattomaan muuttujaan `paikka` liittyvä karkeistamistaulu. Kuvassa 12 esitettyyn `granularity_schema` kuvauksiin liittyy seuraava tulkinta. Riippumattoman muuttujan `paikka` karkeistushierarkian juuri on `maa` (ilmaisee karkeimman karkeistustason). Se jakautuu alueisiin, joka edelleen jakautuu kauppoihin.

Ilmentymätasolla esitetään kaaviotasolla esitetyn käsitteellisen hierarkian eräs todentuma. Tässä tapauksessa riippumattoman muuttujan `paikka` kaaviotason karkein hierarkiataso `maa` kiinnitetään ilmentymätasolla ilmentymään `suomi`.

Edelleen kaaviotason käsitteellinen hierarkiataso `alue` kiinnitetään ilmentymätasolla alueiksi `etelä` ja `itä`, joista siis tässä tietojärjestelmässä koostuu `maa suomi`. Edelleen kaaviotason karkeistamistasoa `alue` vastaa ilmentymätasolla ilmentymät `itä` ja `etelä`, joista siis tässä sovelluksessa koostuu `suomi`.

Karkeistamistaulussa sekä kaaviotaso että ilmentymätaso määrittelevät puun, joiden juurisolmut kuvaavat karkeimman aggregointitason. Nämä kaksi puuta liittyvät toisiinsa niiden rakenteen avulla siten, että kaaviotason puun juuresta lasketun etäisyyden X päässä oleva arvo kuvautuu ilmentymätason puun juuresta lasketulla etäisyydellä X oleviin arvoihin.



Kuva 13: Riippumattoman muuttujan "Paikka" karkeistushierarkian kaavio- ja ilmentymätaso.

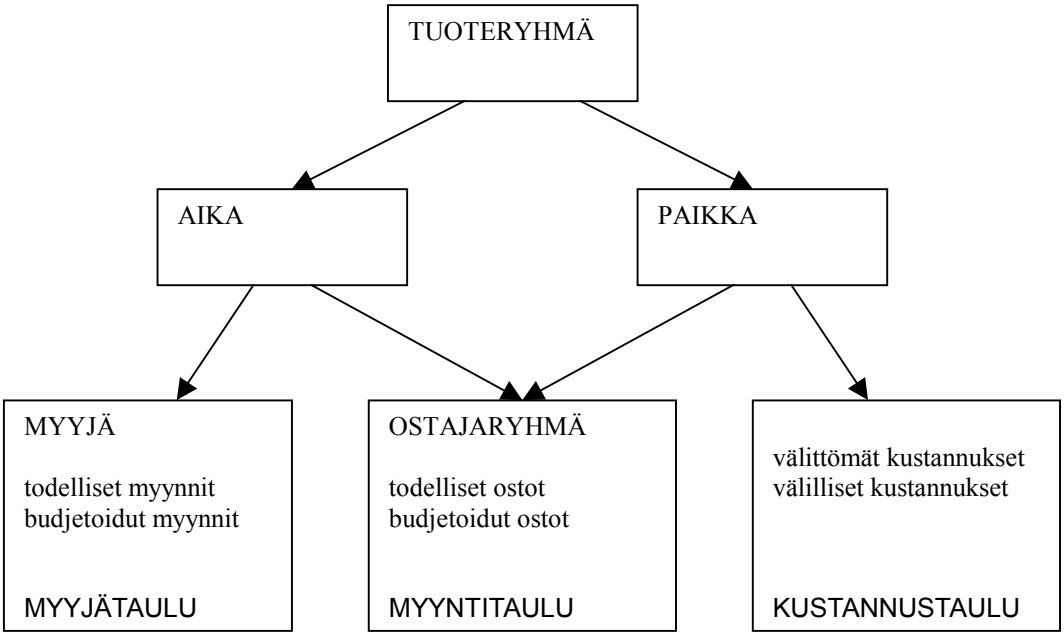
Kuvassa 13 esitellään graafisessa muodossa kuvassa 12 esitetyn karkeistamistaulun kaavio- ja ilmentymätasot. Tästä esityksestä on helppo huomata, miten kaaviotason ja ilmentymätason hierarkiat liittyvät toisiinsa.

5. Esimerkkijärjestelmä

Seuraavaksi esitellään esimerkkijärjestelmä, jota käytetään esimerkkikyselyjen tekemiseen kehitetyllä kyselykielellä. Järjestelmään on määritelty kolme MOLAP –taulua: myyjätaulu, myyntitaulu ja kustannustaulu. MOLAP –kuutioon määritelty myös viisi riippumatonta muuttujaa (tuoteryhmä, aika, paikka, myyjä ja ostajaryhma) ja kuusi riippuvaa muuttujaa (todelliset ja budjetoidut myynnit, todelliset ja budjetoidut ostot ja välittömät ja välilliset kustannukset). Lisäksi on määritelty jokaiseen riippumattomaan muuttujaan liittyvä ominaisuustaulu ja karkeistushierarkia.

5.1. MOLAP –TAULUT

Järjestelmään kuuluvien MOLAP –taulujen muuttujat esitellään kuvassa 14.



Kuva 14: Esimerkkijärjestelmän MOLAP –taulujen riippuvat ja riippumattomat muuttujat.

Kuvan 14 hierarkiassa esitetään taulujen riippumattomat ja riippuvat muuttujat. Kuvaa tulkitaan siten, että esimerkiksi MYYJÄTAULU:n liittyy riippumattomat muuttujat TUOTERYHMÄ, AIKA ja MYYJÄ ja riippuvat muuttujat todelliset myynnit ja budjetoidut myynnit. Kuvasta huomataan, että riippumaton muuttuja TUOTERYHMÄ kuuluu kaikkiin MOLAP –kuutiossa oleviin MOLAP –tauluihin. Lisäksi MYYJÄTAULU ja MYYNTITAULU jakavat riippumattoman muuttujan AIKA. Samoin MYYNTITAULU ja KUSTANNUSTAULU jakavat

riippumatoman muuttujan PAIKKA. Kuvissa 15 – 17 esitetään kukin MOLAP – taulu yksityiskohtaisesti – sisältäen myös ilmentymätason.

			PAIKKA								
			kauppa 1			kauppa 2			kauppa 3		
AIKA	OSTOT	TUOTE-RYHMÄ	nuoret	keski-ikaiset	vanhat	nuoret	keski-ikaiset	vanhat	nuoret	keski-ikaiset	vanhat
1. neljännes	todellinen	elekroniikka	50	500	120	90	254	152	68	345	542
		huonekalut	101	350	250	50	560	425	70	369	152
	budjetoitu	elekroniikka	60	500	120	100	300	250	70	400	600
		huonekalut	100	300	210	50	550	500	70	400	250
2. neljännes	todellinen	elekroniikka	47	460	150	79	354	234	74	245	652
		huonekalut	90	280	210	48	480	325	69	384	254
	budjetoitu	elekroniikka	60	500	120	100	300	250	70	400	600
		huonekalut	100	300	210	50	550	500	70	400	250
3. neljännes	todellinen	elekroniikka	38	530	100	86	265	352	63	512	458
		huonekalut	126	400	220	54	570	354	48	452	235
	budjetoitu	elekroniikka	60	500	120	100	300	250	70	400	600
		huonekalut	100	300	210	50	550	500	70	400	250
4. neljännes	todellinen	elekroniikka	60	600	90	110	452	354	80	584	689
		huonekalut	86	365	190	60	600	652	90	654	245
	budjetoitu	elekroniikka	60	500	120	100	300	250	70	400	600
		huonekalut	100	300	210	50	550	500	70	400	250

Kuva 15: Myyntitaulu.

Myyntitaulussa (kuva 15) kuvataan todelliset ja budjetoidut ostot ryhmiteltynä ajan, paikan, tuoteryhmän ja ostajaryhmän mukaan.

		TUOTERYHMÄ	
PAIKKA	KUSTANNUKSET	elektroniikka	huonekalut
kauppa 1	välittömät	20	50
	välilliset	30	40
kauppa 2	välittömät	15	70
	välilliset	20	50
kauppa 3	välittömät	30	40
	välilliset	20	45

Kuva 16: Kustannustaulu.

Kustannustaulussa (kuva 16) kuvataan välittömät ja välilliset kustannukset ryhmiteltynä paikan ja tuoteryhmän mukaan.

		MYYJÄ							
		myyja 1		myyja 2		myyja 3		myyja 4	
AIKA	MYYNNIT	huone-kalut	elektro-niikka	huone-kalut	elektro-niikka	huone-kalut	elektro-niikka	huone-kalut	elektro-niikka
1. Neljännes	todellinen	300	300	401	370	1035	496	591	955
	budjetoitu	300	300	310	380	1100	650	720	1070
2. Neljännes	todellinen	250	300	330	357	853	667	707	971
	budjetoitu	300	300	310	380	1100	650	720	1070
3. Neljännes	todellinen	350	330	396	338	978	703	735	1033
	budjetoitu	300	300	310	380	1100	650	720	1070
4. Neljännes	todellinen	320	200	321	550	1312	916	989	1353
	budjetoitu	300	300	310	380	1100	650	720	1070

Kuva 17: Myyjätaulu.

Myyjätaulussa esitetään todelliset ja budjetoidut myynnit ajan, tuoteryhmän ja myyjän mukaan ryhmiteltynä.

5.2. OMINAISUUSTAULUT

Seuraavaksi esitellään järjestelmässä olevat ominaisuustaulut. Kuvassa 18 esitetään ominaisuustaulun `ostajaryhmat` ilmentymätaso.

ostajaryhmat	ostajaryhma	alkuika	loppuika
	nuoret	0	25
	keski_ikaiset	26	50
	vanhat	51	100

Kuva 18: Ominaisuustaulu 'ostajaryhmat'.

Ominaisuustauluun `ostajaryhmat` liittyvät attribuutit `ostajaryhma`, `alkuika` ja `loppuika`. Ensimmäinen attribuutti (`ostajaryhma`) on erikoisasemassa, koska se on riippumaton muuttuja, johon muut attribuutit liittyvät. Ostajaryhmiä ovat `nuoret`, `keski_ikaiset` ja `vanhat`, joihin kuhunkin liittyy attribuuttien `alkuika` ja `loppuika` arvot. Nämä arvot määrittelevät kunkin ostajaryhmän rajat ikävuosissa.

Seuraavaksi esitellään ominaisuustaulu `ostajaryhmien_koot` (kuva 19).

ostajaryhmien_koot	paikka	nuoria	keski_ikaisia	vanhoja
	kauppa1	100	250	70
	kauppa2	70	220	50
	kauppa3	90	150	100

Kuva 19: Ominaisuustaulu 'ostajaryhmien_koot'.

Ominaisuustauluun `ostajaryhmien_koot` liittyy attribuutit `paikka`, `nuoria`, `keski_ikaisia` ja `vanhoja`. Taulun riippumaton muuttuja on muuttuja `paikka`,

jolla on arvot `kauppa1`, `kauppa2` ja `kauppa3`. Arvoihin liittyy attribuuttien `nuoria`, `keski_ikaisia` ja `vanhoja` arvot, jotka ilmaisevat kyseiseen ostajaryhmään kuuluvien ihmisten lukumäärän tietyn kaupan asiakaskunnassa.

Kuvassa 20 esitellään ominaisuustaulu `neljannesten_ajat`.

neljannesten_ajat	aika	alkukuukausi	loppukuukausi
	ensimmainen	1	3
	toinen	4	6
	kolmas	7	9
	neljas	10	12

Kuva 20: Ominaisuustaulu 'neljannesten_ajat'.

Ominaisuustaulussa `neljannesten_ajat` (kuva 20) ilmaistaan kuhunkin riippumattoman muuttujan `aika` arvoon (`ensimmainen`, `toinen`, `kolmas` ja `neljas`) liittyvä attribuutti `alkukuukausi` ja `loppukuukausi`. Nämä attribuuttien arvot ilmaisevat sen, mitkä kuukaudet kuuluvat mihinkin neljännekseen.

Kuvassa 21 esitellään ominaisuustaulu `artikkelit`, joka liittyy riippumattomaan muuttujaan `tuoteryhma`.

artikkelit	tuoteryhma	artikkelien_maara
	huonekalut	45
	elektroniikka	26

Kuva 21: Ominaisuustaulu 'artikkelit'.

Ominaisuustaulu artikkelit ilmaisee sen, kuinka monta eri artikkelia kuuluu kuhunkin tuoteryhmään. Tuoteryhmiä ovat `huonekalut` ja `elektroniikka`.

Kuvassa 22 esitellään riippumattomaan muuttujaan `myyja` liittyvä ominaisuustaulu `myyjien_tiedot`.

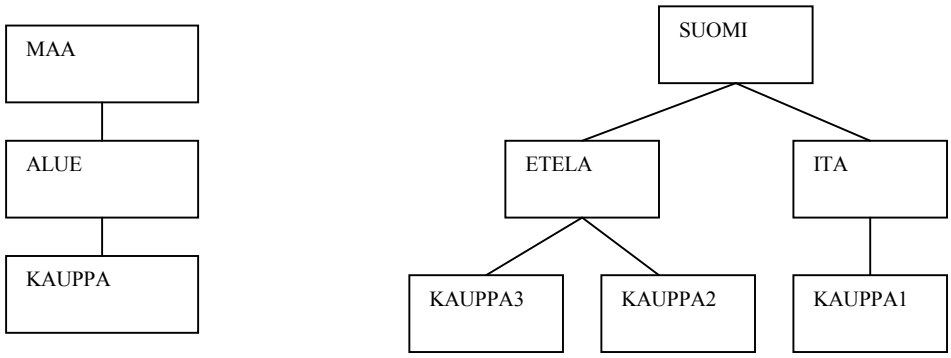
myyjien_tiedot	myyja	nimi	osoite	palkka
	yksi	arttu	kotikatu	15000
	kaksi	liisa	puistokatu	15000
	kolme	leena	rantatie	9000
	nelja	mauno	keskuskatu	10000

Kuva 22: Ominaisuustaulu 'myyjien tiedot'.

Jokaiseen riippumattoman muuttujan `myyjä` arvoon liitetään myyjään liittyvä `nimi` `osoite` ja `palkka`.

5.3. KARKEISTUSHIERARKIAT

Seuraavaksi esitellään riippumattomen muuttujien arvoihin liittyvät karkeistushierarkiat. Ensimmäiseksi esitellään riippumattomaan muuttujaan `paikka` liittyvä karkeistushierarkia (kuva 23). Hierarkia on visualisoitu jo kuvassa 13, mutta yhtenäisyyden vuoksi se esitellään tässä uudestaan.



Kuva 23: Riippumattomaan muuttujaan paikka liittyvä karkeistushierarkia.

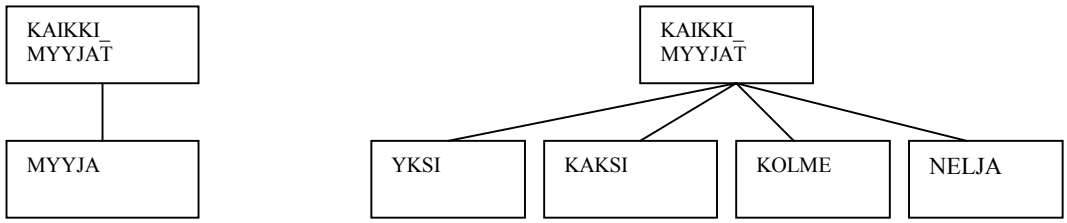
Kuvassa vasemmalla esitellään hierarkian kaaviotaso ja oikealla ilmentymätaso. Kaaviotasolla MAA voidaan jakaa komponentteihin ALUE, jotka edelleen voidaan jakaa komponentteihin KAUPPA. Kaaviotason MAA liittyy ilmentymätasolla ilmentymään SUOMI. SUOMI voidaan jakaa komponentteihin ETELA ja ITA (kaaviotason taso ALUE), joista ETELA voidaan jakaa KAUPPA2:n ja KAUPPA3:n (kaaviotasolla taso KAUPPA). Alue ITA vastaa KAUPPA1:stä.

Seuraavaksi esitellään riippumattomaan muuttujaan `tuoteryhma` liittyvä karkeistushierarkia (kuva 24).

Kuva 24: Riippumattomaan muuttujaan tuoteryhmä liittyvä karkeistushierarkia.

Karkeistushierarkiassa on kaksi tasoa. Kaaviotasolla tasot ovat KAIKKI_TUOTTEET, joka voidaan jakaa tasolle TUOTE. Ilmentymätasolla ilmaistaan, että KAIKKI_TUOTTEET koostuvat tuoteryhmistä HUONEKALUT ja ELEKTRONIIKKA.

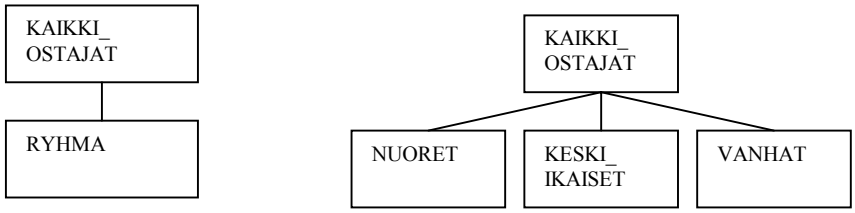
Kuvassa 25 esitellään riippumattomaan muuttujaan `myyja` liittyvä karkeistushierarkia.



Kuva 25: Riipppumattomaan muuttujaan myyja liittyvä karkeistushierarkia.

Kaavio- ja ilmentymätason juuret ovat samannimiset, KAIKKI_MYYJAT. Kaaviotasolla se voidaan jakaa tasolle MYYJA. Ilmentymätasolla KAIKKI_MYYJAT jakautuu yksittäisiksi myyjiksi YKSI, KAKSI, KOLME ja NELJA.

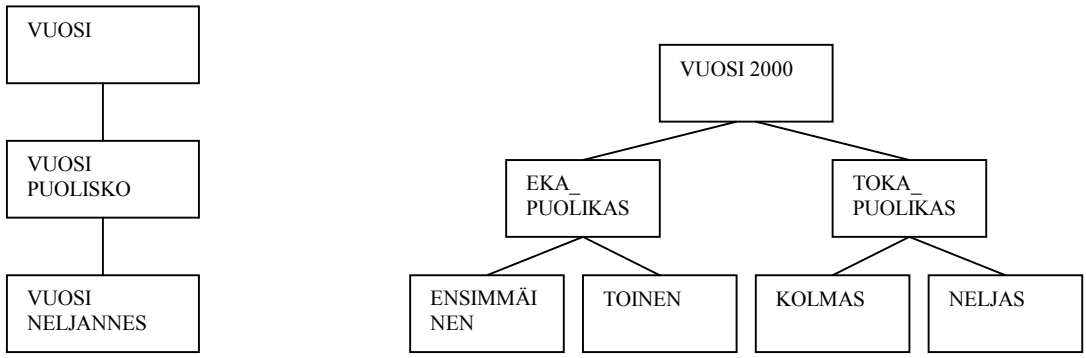
Kuva 26 esittelee riippumattomaan muuttujaan `ostajaryhma` liittyvän karkeistushierarkian.



Kuva 26: Riippumattomaan muuttujaan ostajarhma liittyvä karkeistushierarkia.

Kaaviotason juuri KAIKKI_OSTAJAT voidaan jakaa tasolle RYHMA. Ilmentymätasolla juuri KAIKKI_OSTAJAT voidaan jakaa osiin NUORET, KESKI_IKAISET ja VANHAT.

Kuva 27 visualisoi riippumattomaan muuttujaan aika liittyvän karkeistushierarkian.



Kuva 27: Riippumattomaan muuttujaan aika liittyvä karkeistushierarkia.

Kaaviotason juuri VUOSI vastaa ilmentymätason juurta VUOSI 2000. VUOSI voidaan jakaa tasolle VUOSIPUOLISKO, ja VUOSI 2000 voidaan jakaa vuosipuoliskoiksi EKA_PUOLIKAS ja TOKA_PUOLIKAS: VUOSIPUOLISKO voidaan edelleen jakaa tasolle VUOSINELJANNES, jota vastaavat arvot ilmentymätasolla ovat ENSIMMAINEN, TOINEN, KOLMAS ja NELJAS.

6. Näkymät

Tärkein OLAP:n tarjoama muutos perinteisiin tietokantajärjestelmiin verrattuna on näkymien muodostamisen joustavuus. Näkymällä tarkoitetaan näkemystauluja, jotka muodostetaan kantatauluista. Kuten kappaleessa 1.3 todettiin, joustavia näkymiä ei voidan tuottaa perinteisillä menetelmillä, kuten taulukkolaskennalla ja SQL:llä. Tässä luvussa esitellään OLAP:n perusoperaatiot, joilla näkymiä muodostetaan. Kehitetyllä view –operaatiolla on mahdollista konstruoida monipuolisia näkymiä ja sillä on mahdollista esittää olemassa olevien OLAP -järjestelmien perusoperaatiot.

6.1. VIEW –OPERAATIO

View –operaattori luo uuden näkymän kantatauluista perustuen käyttäjän määrittelemään muotoon. Operaatiolle annetaan kaksi argumenttia. Ensimmäinen argumentti (muodoltaan järjestetty joukko) kuvaa luotavan näkymän rakenteen. Toinen argumentti ilmaisee näkymässä olevien komponenttien (sarakkeiden) muodostamisperiaatteet.

```
view(new_view_name(muuttuja_nimi_1, … , muuttuja_nimi_n,
muuttuja_nimi_n+1, … , muuttuja_nimi_m),
[new_view_dim(muuttuja_nimi_n+1, muuttujan_nimi_x,
[muuttujan_x_arvot], riippuva_muuttuja_a), … ,
new_view_dim(muuttuja_nimi_m, muuttujan_nimi_y,
[muuttujan_y_arvot], riippuva_muuttuja_b)]).
```

Kuva 28: Yleinen view- operaatio.

Kuvassa 28 esitellään, mistä komponenteista view –operaatio koostuu. Siinä `new_view_name` on luotavan näkemystaulun nimi ja `muuttuja_nimi_1, … , muuttuja_nimi_n` ovat joko olemassa olevien riippumattomien muuttujien nimiä tai niille määriteltyjen karkeusasteiden nimiä. `Muuttuja_nimi_n+1, … , muuttuja_nimi_m` ovat sarakkeita, joiden muodostaminen kuvataan toisessa argumentissa.

Toinen argumentti koostuu `new_view_dim` -nimisistä komponenteista (muodoltaan järjestettyjä joukkoja). Jokainen komponentti määrittelee yhden tulostaulun sarakkeen. Komponentin ensimmäisenä argumenttina on luotavassa näkemyksessä

olevan muuttujan nimi, jonka käyttäjä nimeää. Komponentin toinen argumentti (`muuttujan_nimi_x` ja `muuttujan_nimi_y` kuvassa 28) viittaa OLAP –sovelluksessa olevaan riippumattomaan muuttujaan, jota käytetään näkemyksessä mukana olevan muuttujan konstruointiin. Usea sarake voi jakaa saman riippumattoman muuttujan. Kolmannessa argumentissa ([`muuttujan_x_arvot`] ja [`muuttujan_y_arvot`] kuvassa 28) ilmaistaan, mitkä toisessa argumentissa annetun riippumattoman muuttujan arvot muodostavat komponentin 1. argumentissa ilmaiseman muuttujan arvot. Näkemystaulun eri sarakkeiden määrittelyssä voidaan käyttää samoja tietyn riippumattoman muuttujan arvoja. Viimeinen argumentti ilmaisee mielenkiinnon kohteena olevan riippuvan muuttujan (`riippuva_muuttuja_a` ja `riippuva_muuttuja_b` kuvassa 28). Usea sarake voi jakaa saman riippuvan muuttujan.

View- operaatio on toteutettu siten, että se suorittaa automaattisen navigoinnin sovellutuksen MOLAP-taulujen välillä. Loppukäyttäjälle tämä navigointi ei näy mitenkään ja hän voi tehdä kyselyitä tietämättä, että tietoja haetaan useammasta taulusta. Toinen merkittävä ominaisuus on jokaisen sarakkeen riippuvan muuttujan erillinen määrittely. Tämä mahdollistaa useiden eri riippuvien muuttujien tarkastelun samassa näkymässä.

Oletetaan, että esimerkissämme loppukäyttäjä haluaa tutkia tuoteryhmien välittömiä kustannuksia alueittain ryhmiteltynä (ks. kuva 16). Tuoteryhminä ovat siis `huonekalut` ja `elektroniikka` ja alueina ovat `kauppa1`, `kauppa2` ja `kauppa3`. Riippumatoman muuttujan `tuoteryhmä` arvot halutaan ensimmäiselle sarakkeelle, ja `alueet` määrittelemään sarakkeita, joilla kuvataan riippuvan muuttujan välittömät kustannukset arvoja. Tämän kyselytarpeen tyydyttävä kysely esitetään view –operaatiolla kuvassa 29.

```
view(normal(tuoteryhma, kauppa1_val, kauppa2_val, kauppa3_val),
[new_view_dim(kauppa1_val, paikka, [kauppa1], valittomat_kust),
new_view_dim(kauppa2_val, paikka, [kauppa2], valittomat_kust),
new_view_dim(kauppa3_val, paikka, [kauppa3], valittomat_kust)]).
```

Kuva 29: Esimerkkikysely käyttäen view –operaatiota.

Kyselyssä käyttäjä antaa näkymätaulun nimeksi `normal`. View –operaation ensimmäisessä argumentissa ilmaistaan luotavan näkemystaulun sarakkeet

tuoteryhma, kauppa1_val, kauppa2_val ja kauppa3_val. Tuoteryhma on riippumattoman muuttujan nimi, joten sen arvot elektroniikka ja huonekalut esiintyvät tuoteryhma –sarakkeen arvoina. Toista, kolmatta ja neljättä saraketta ei sellaisenaan vastaa suoraan mikään OLAP –sovelluksessa oleva muuttuja. Siksi ne on määriteltävä view –operaation toisessa argumentissa.

Toisena argumenttina olevassa listassa on kolme alkiota. Ensimmäinen määrittelee sarakkeen kauppa1_val. Se liittyy riippumattoman muuttujan paikka arvoon kauppa1. Sarakkeen arvot ovat siis kauppa1:n liittyvän riippuvan muuttujan valittomat_kust (välittömät kustannukset) arvoja. Toinen ja kolmas alkio määrittelevät atomit kauppa2_val ja kauppa3_val. Nämä alkiot määritellään analogisesti verrattuna kauppa1_val –alkion määrittelyyn. Kyselyn tulos esitetään kuvassa 30 visualisoituna.

normal	tuoteryhma	kauppa1_val	kauppa2_val	kauppa3_val
	elektroniikka	20	15	30
	huonekalut	50	70	40

Kuva 30: Esimerkkikyselyn tulos.

Ensimmäinen rivi tuloksessa ilmaisee näkemystaulun kaaviotason, joka esitetään samalla tavalla kuin MOLAP –taulujen kaaviotaso MOLAP –kuutiossa. Loput kaksi riviä ilmaisevat näkemystaulun ilmentymätason. Ilmentymätason ensimmäinen rivi ilmaisee, että elektroniikasta on koitunut kauppa1:ssä välittömiä kustannuksia yhteensä 20 yksikköä, kauppa2:ssa 15 yksikköä ja kauppa3:ssa 30 yksikköä. Toinen rivi ilmaisee vastaavat luvut tuoteryhmälle huonekalut.

6.2. OLAP:N PERUSOPERAATIOT

Edellä esitelty view –operaatio on erittäin ilmaisuvoimainen ja sillä on mahdollista suorittaa nykyisten OLAP –järjestelmien tyypilliset operaatiot kuten *projektio*, *konkatenaatio*, *porautuminen* (drill up/down), *pyöristäminen* (roll up/down) ja *kääntäminen* (pivoting) [Tho97, Dat00, AGS97]. View –operaatiolla kyetään ilmaisemaan myös mikä tahansa perusoperaatioiden kombinaatio. Seuraavaksi tarkastellaan näiden perusoperaatioiden esittämistä view –operaatiolla.

6.2.1. Projektio

Projektio –operaation avulla käyttäjä valitsee vain osan kantataulun muuttujista luotavaan näkemystauluun. Tällöin näkemystaulun sanotaan olevan projektio

kantataulusta. Projektio –operaatiolla aikaansaadaan pelkistetympiä näkemystauluja, kun tarkastelun kannalta epäolennainen tieto poistetaan näkemyksestä.

Projektio –operaatiolla on kaksi käyttötapaa. Ensiksi sen avulla valitaan ne riippumattomat ja riippuvat muuttujat, jotka halutaan tulostauluun. Toiseksi sen avulla voidaan valita riippumattomien muuttujien arvoista tulostauluun ne, jotka ovat analysoinnin kannalta kiinnostavia. Tämä tapa soveltaa projektio –operaatiota, on hyvin lähellä valinta (select) –operaatiota. Projektion määrittelyssä siis view –operaation molemmat argumentit ovat olennaisia.

```
view(projection(tuoteryhma, kauppa1_val, kauppa2_val),
[new_view_dim(kauppa1_val, paikka, [kauppa1], valittomat_kust),
new_view_dim(kauppa2_val, paikka, [kauppa2], valittomat_kust)]).
```

Kuva 31: View –operaatio, joka suorittaa projektio –operaation.

Kuvassa 31 esitellään kysely, joka suorittaa projektio –operaation kustannustaulun suhteen (ks. kuva 16). Tässä esimerkissä käyttäjä haluaa tulostauluun riippumattomat muuttujat `tuoteryhmä` ja `paikka`. Riippumattomaksi muuttujaksi käyttäjä haluaa `valittomat_kust` (välittömät kustannukset). Välittömät kustannukset ja sitä määrittelevät riippumattomat muuttujat `tuoteryhmä` ja `paikka` (ks. kuva 16) kuuluvat kustannustauluun. Yhtään riippumatonta muuttujaa siis ei jätetä pois tulostaulusta. Kustannustaulussa riippuvina muuttujina ovat välilliset (`valilliset_kust`) ja välittömät kustannukset, joista siis vain välittömät kustannukset valitaan tulostauluun.

Jokainen view –operaation toisessa argumentissa oleva muuttuja määrittelee yhden sarakkeen tulostauluun. Esimerkissä ensimmäiseen sarakkeeseen liittyy riippumattoman muuttujan paikka arvo `kauppa1`, ja toiseen `kauppa2`. Tässä tarkastelussa on jätetty riippumattoman muuttujan `paikka` arvo `kauppa3` huomiotta kyselyn kannalta kiinnostamattomana. Operaation tulos esitetään kuvassa 32.

projection	tuoteryhma	kauppa1_val	kauppa2_val
	elektroniikka	20	15
	huonekalut	50	70

Kuva 32: Esimerkkitaulun eräs projektio.

6.2.2. Konkatenaatio

Konkatenaatio tarkoittaa kahden tai useamman kanta- tai näkemystaulun liittämistä yhteen yhdeksi näkemystauluksi. Konkatenaatio tapahtuu kanta- tai näkemystaulujen yhteisten riippumattomien muuttujien avulla. Tämän operaation avulla voidaan tutkia eri tauluissa olevia riippuvien muuttujien arvoja niihin liittyvien yhteisten riippumattomien muuttujien arvojen avulla. Nämä yhteiset riippumattomat muuttujat ovat ainoita riippumattomia muuttujia, joiden avulla tiedot voidaan semanttisesti yhdistää eri kantatauluista.

```
view(concatenation(tuoteryhma, kauppa1_val, kauppa1_tod),
[new_view_dim(kauppa1_val, paikka, [kauppa1], valittomat_kust),
new_view_dim(kauppa1_tod, paikka, [kauppa1],
todelliset_ostot)]).
```

Kuva 33: View –operaatio, joka suorittaa konkatenaation

Kuvassa 33 esitetään view –operaatio, joka sisältää konkatenaatio –operaation. Tässä kyselyssä tarvitaan tietoa sekä kustannustaulusta että myyntitaulusta. Ensimmäiselle sarakkeelle on valittu riippumaton muuttuja `tuoteryhmä`, jonka siis kaikki MOLAP –taulut jakavat (ks. kuva 14). Riippuvia muuttujia kuvailevia sarakkeita määrittelee riippumaton muuttuja `paikka`. Toisen riippuvia muuttujia kuvaavan sarakkeen ensimmäisestä sarakkeesta poikkeava tulkinta on saatu muuttamalla tarkasteltavaa riippuvaa muuttujaa view –operaation toisen argumentin viimeisessä alkiossa välittömistä kustannuksista todellisiin ostoihin. Operaation tuottama tulostaulu esitetään kuvassa 34.

concatenate	tuoteryhma	kauppa1_val	kauppa1 _tod
	huonekalut	50	2668
	elektroniikka	20	2745

Kuva 34: Tulostaulu kuvan 33 view –operaatiolle.

6.2.3. Porautuminen

Porautuminen voi tapahtua joko alaspäin (Drill-Down) tai ylöspäin (Drill-Up). Porautumisessa siirrytään tietojen karkeistamishierarkiassa ylös tai alaspäin, jolloin muuttujan arvot tulevat yksityiskohtaisemmiksi tai karkeammiksi. Porautuminen on mahdollista vain muuttujille, joille on määritelty karkeistamishierarkia. Esimerkissä MOLAP-kuutiossa (ks. kuva 14) on riippumaton muuttuja `aika`, jonka karkeistushierarkia käsittää tasot `vuosi`, `puolivuodet` ja `neljännesvuodet` (kuva 27). Esimerkiksi siirryttäessä tarkastelemaan aikaa puolivuosien tasolta neljännesvuosien tasolle poraudutaan alaspäin. Jos taas tarkastelutasoa muutetaan neljännesvuosista puolivuosiksi, poraudutaan ylöspäin. Porautuminen tapahtuu aina sovellukseen luotujen karkeistustasojen perusteella. Kuten aikaisemmin jo mainittiin view –operaation ensimmäisen argumentin alkoina voi olla joko riippumattomien muuttujien nimiä tai niille määriteltyjen karkeistustasojen nimiä. Jälkimmäisessä tapauksessa view –operaatio porautuu annetulle karkeistustasolle. Tämänkaltainen porautuminen on mahdollista vain sarakkeilla esitetyille riippumattomille muuttujille. Haluttaessa porautua riippuvien muuttujien sarakkeita määrittelevien riippumattomien muuttujien karkeistustasoille, pitää taso manuaalisesti asettaa view –operaation toisen argumentin komponenttien 3. argumenttina oleviin listoihin (`[muuttujan_x_arvot]` ja `[muuttujan_y_arvot]` kuvassa 28).

```
view(drill(kaikki_tuotteet, kauppa1_val, kauppa2_val,
kauppa3_val), [new_view_dim(kauppa1_val, paikka, [kauppa1],
valittomat_kust), new_view_dim(kauppa2_val, paikka, [kauppa2],
valittomat_kust), new_view_dim(kauppa3_val, paikka, [kauppa3],
valittomat_kust)]).
```

Kuva 35: View –operaatio, joka suorittaa porautumis –operaation.

Kuvassa 35 annetaan view –operaatio, joka sisältää porautumisen. Tässä on porauduttu riippumattoman muuttujan `tuoteryhmä`, karkeistustasolle `kaikki_tuotteet` (ks kuva 24). Tähän kuuluvat tuoteryhmät `elektroniikka` ja `huonekalut`. Operaation tulos annetaan kuvassa 36.

drill	kaikki_tuotteet	kauppa1_val	kauppa2_val	kauppa3_val

kaikki_tuotteet	70	85	70

Kuva 36: Kuvassa 35 esitetyn view –operaation tuottama tulostaulu.

6.2.4. Pyöristäminen

Pyöristäminen (Roll-up) on samantapainen operaatio kuin porautuminen. Porautumisessa liikuttiin riippumattomalle muuttujalle määritellyssä karkeistamishierarkiassa kun taas pyöristämistä sovelletaan niille riippumattomille muuttujille, joille joko ei ole määritelty karkeistamishierarkoita tai joille halutaan luoda karkeistamishierarkiasta poikkeava karkeistustaso. Pyöristettäessä siis luodaan uusi karkeistustaso muuttujalle sen sijaan, että käytettäisiin jo olemassa olevia kategorioita [Dat00]. Esimerkiksi jos esimerkkiympäristössä riippumattoman muuttujan `aika` perusteella halutaan jotakin rippuvaa muuttujaa tutkia ¾ vuosittain, ja sen normaalissa karkeistushierarkiassahierarkiassa ei ole määriteltynä ¾ vuosia (ks. kuva 27). Tällöin joudutaan määrittelemään uusi karkeistustaso, jolle poraudutaan. Pyöristäminen voidaan suorittaa vain sarakkeita määritteleville riippumattomille muuttujille. Pyöristämisessä määriteltävä uusi karkeistustaso määritellään view –operaation toisessa argumentissa.

```
view(roll(tuoteryhma, kauppa1ja2_val, kauppa3_val),
[new_view_dim(kauppa1ja2_val, paikka, [kauppa1, kauppa2],
valittomat_kust), new_view_dim(kauppa3, paikka, [kauppa3],
valittomat_kust)]).
```

Kuva 37: Pyöristämisoperaation suorittava view –operaatio.

Esimerkissä (kuva 37) on esitelty eräs alkuperäiselle taululle (kuva 30) suoritetun pyöristämis –operaation lopputulos. Tässä taulussa ryhmitellään `kauppa1` ja `kauppa2` samaan sarakkeeseen, ja `kauppa3` omaansa. Tämä ryhmittely poikkeaa järjestelmään valmiiksi määritellyistä karkeistustasoista (ks. kuva 23). Operaation tulos esitetään kuvassa 38.

roll	tuoteryhmä	kauppa1ja2_val	kauppa3_val
	elektroniikka	35	30
	huonekalut	120	40

Kuva 38: Kuvan 37 view –operaation tuottama tulostaulu.

6.2.5. Kääntäminen

Kääntäminen tarkoittaa operaatiota, joka vaihtaa tietyssä näkemyksessä olevia riippumattomia muuttujia siten, että taulussa olevan informaation ensisijainen

organisointiperiaate muuttuu. Kääntäminen voi tapahtua joko kaikille tai vain osalle taulun ulottuvuuksista. Näkymän uudelleenstrukturointi voi tuoda joitakin riippuvuus –suhteita paremmin näkyville. Kääntämisoperaatio on toteutettavissa view –operaatiolla muuttamalla riippumattomien muuttujien määrittelyjen paikkaa view –operaation ensimmäisen ja toisen argumentin välillä.

```
view(pivot(paikka, el_val, hu_val), [new_view_dim(el_val,
tuoteryhma, [elektroniikka], valittomat_kust),
new_view_dim(hu_val, tuoteryhma, [huonekalut],
valittomat_kust)]).
```

Kuva 39: View –operaatio, joka kuvaa kääntämis –operaation ksutannustaulun suhteen (kuva 16).

Edellä olevissa kyselyissä (kuvat 29 – 38) riippumattoman muuttujan `tuoteryhmä` arvot esiintyvät sarakkeessa, kun taas riippumatoman muuttujan `paikka` arvoja on käytetty ryhmittelemään riippuvien muuttujien arvoja muissa sarakkeissa. Kuvassa 39 esitetyssä esimerkissä `paikka` sen sijaan määritellään ensimmäisessä argumentissa ensisijaiseksi organisointiperiaatteeksi, jolloin sen arvot esiintyvät sarakkeessa. Sen sijaan riippumattoman muuttujan `tuoteryhmä` arvoja (ks. toinen argumentti) käytetään ryhmittelemään sarakkeissa riippuvan muuttujan `valittomat_kust` arvoja.

pivot	paikka	el_val	hu_val
	kauppa1	20	50
	kauppa2	15	70
	kauppa3	30	40

Kuva 40: Kuvan 39 view –operaation tuottama tulostaulu.

Kuvassa 40 esitetään kääntämisoperaation tuottama esimerkkitaulu perustuen kuvassa 30 olevaan tauluun (joka itsessään on luotu näkymä). Kaaviotason sarakkeen `paikka` arvoina ilmentymätasolla ovat mahdolliset kaupat. Sarakkeiden `el_val` ja `hu_val` arvoina esitetään tuoteryhmiin `elektroniikka` ja `huonekalut` liittyvät välittömät kustannukset.

7. Add -operaatio

Usein tulostauluihin halutaan liittää myös johdettuja ja aggregoituja muuttujia. Prototyypissä tämä mahdollisuus on annettu luomalla add –operaatio, joka lisää tulostauluun haluttua informaatiota. Add -operaatiolle annetaan yksi argumentti, joka on muodoltaan järjestämätön joukko. Joukon komponentteina on yksipaikkaisia yhdistettyjä termejä, joiden niminä käytetään joitakin MOLAP –taulun laajennusoperaatioiden nimiä. Argumenttina näillä komponenteilla on taulun nimi, johon lisätietoa halutaan liittää. Add –operaation rakenne esitellään kuvassa 41.

```
add([Operation_name_1(Table_name_1), … ,
Operation_name_n(Table_name_n)]).
```

Kuva 41: Add –operaatio.

Add –operaatiossa olevat `Operation_name_1, … , Operation_name_n` viittaavat prototyypissä toteutettuihin MOLAP –taulun laajennusoperaatioihin. `Table_name_1, … , Table_name_n` ovat MOLAP –kuutiossa olevia MOLAP –tauluja. Useita laajennusoperaatioita voidaan kohdistaa samaan tauluun. Add –operaatiossa edellytytetään, että taulu, johon lisäinformaatiota halutaan, on olemassa.

Prototyypissä toteutetut MOLAP –taulujen laajennusoperaatiot ovat `row_sums`, `col_sums`, `row_avg`, `col_avg` ja `divide(X,Y)`. `Row_sums` –operaatio laskee rivien kokonaissummat, ja lisää tämän tiedon ilmaisevan sarakkeen näkemystauluun. `Col_sums` –operaatio laskee sarakkeiden kokonaissummat, ja lisää tämän tiedon ilmaisevan rivin näkemystauluun. Operaatiot `row_avg` ja `col_avg` laskevat rivien ja sarakkeiden keskiarvot. `Divide(X,Y)` –operaatio on tarkoitettu suhteiden laskemiseksi (johdettu muuttuja). Operaatiolle annetaan kaksi argumenttia `X` ja `Y`, jotka ovat kokonaislukuja. Nämä argumentit ilmaisevat minkä sarakkeen (`X`) arvot jaetaan minkä sarakkeen (`Y`) arvolla. Add –operaatiota tullaan myöhemmin soveltamaan esimerkkikyselyjen yhteydessä.

8. Kyselyjen tekeminen kehitetyllä kyselykielellä

Tässä luvussa sekä demonstroidaan kyselyjen tekemistä em. operaattorien avulla että tarkastellaan yhteyden muodostamista MOLAP-taulujen ja relaatioina esitettyjen ominaisuustaulujen välille. MOLAP –taulujen käsittelemiseksi esiteltiin luvussa 5 view –operaatio. MOLAP –taulun laajennusoperaatioiden käsittelemiseksi esiteltiin luvussa 6 operaattori add. Koska view –operaatio tuottaa tuloksena näkemystauluja, jotka ovat rakenteellisesti samankaltaisia kuin MOLAP –taulut, on operaatioiden ketjutus mahdollista. Esimerkki operaatioiden ketjuttamisesta annetaan myöhemmin kohdassa 7.3. View –operaatiolla on mahdollista määritellä myös useita edellä tarkasteltuja perusoperaatiota samalla kertaa.

8.1. USEAMPI OPERAATIO YHDESSÄ VIEW –OPERAATIOSSA

Esimerkkikyselyssä 1 (kuva 42) view –operaatiota sovelletaan siten, että sillä ilmaistaan perusoperaatiot konkatenaatio, projektio ja pyöristäminen.

```
view(conc_roll_proj(tuoteryhma, myyjat1ja2, myyjat3ja4,
kaikki_ostot), [new_view_dim(myyjat1ja2, myyja, [yksi, kaksi],
todellinen_myynti), new_view_dim(myyjat3ja4, myyja, [kolme,
nelja], todellinen_myynti), new_view_dim(kaikki_ostot,
ostajaryhma, [nuoret, keski_ikaiset, vanhat],
todelliset_ostot)]).
```

Kuva 42: Esimerkkikysely 1.

Esimerkkikyselyssä 1 näkemystauluun yhdistetään tietoja myyjätaulusta ja myyntitaulusta (ks. kuvat 14, 15 ja 17). Toinen ja kolmas sarake liittyy myyjätaulussa olevaan riippuvaan muuttujaan `todellinen_myynti`. Esimerkissä suoritetaan pyöristämis –operaatio, jossa myyjät jaetaan kahteen ryhmään: myyjat `yksi` ja `kaksi` omaansa, ja myyjät `kolme` ja `nelja` omaansa. Tämä jaottelu eroaa sovellutuksessa määritellystä karkeistushierarkiasta (ks. kuva 25). Neljännellä sarakkeella on määriteltynä atomi `kaikki_ostot`. Siihen liittyy riippumattoman muuttujan `ostajaryhma` kaikki arvot (ks. kuva 26). Riippuvana muuttujana neljännellä sarakkeella on `todelliset_ostot`. `Todelliset_ostot` on määritelty myyntitaulussa. (ks. kuva 15). Kysely on esimerkki kyselystä, joka kohdistuu useaan MOLAP –tauluun. Esimerkkikysely osoittaa myös sen, että käyttäjän ei tarvitse kyselyissään ilmaista taulujen välistä navigointia millään

tavalla. Kyselyn lopputuloksesta on jätetty pois joitakin myyjä- ja myyntitaulussa olevia riippumattomia ja riippuvia muuttujia. Toisin sanoen kyselyssä ilmaistaan myös projektio.

conc_roll_proj	tuoteryhma	myyjat1ja2	myyjat3ja4	kaikki_ostot
	huonekalut	2668	7200	9868
	elektroniikka	2745	7094	9839

Kuva 43: Esimerkkikyselyn 1 lopputulos.

Esimerkkikyselyssä 2 (kuva 44) sovelletaan add –operaatiota. Siinä tulostauluun yhdistetään tietoa jokaisesta kolmesta MOLAP –taulusta (kuvat 15 – 17).

```
view(conc_sum(tuoteryhma, myyja1_myynnit, nuorten_ostot,
valitt_kustannukset), [new_view_dim(myyja1_myynnit, myyja,
[yksi], todellinen_myynti), new_view_dim(nuorten_ostot,
ostajaryhma, [nuoret], todelliset_ostot),
new_view_dim(valitt_kustannukset, paikka, [kauppa1, kauppa2,
kauppa3], valittomat_kust)]), add([col_avg(conc_sum)]).
```

Kuva 44: Esimerkkikysely 2.

Tässä kyselyssä käyttäjä haluaa tulostauluun `myyja1`:n todelliset myynnit, `nuorten_ostot` ja kaikki välittömät kustannukset eriteltynä tuoteryhmittäin. Lisäksi käyttäjä haluaa lisätä tulostauluun aggregoidun muuttujan `col_avg` arvot. Sarakkeiden keskiarvo lisätään tulostauluun alimmaksi riviksi (ks. kuva 45).

conc_sum	tuoteryhma	myyja1_myynnit	nuorten_ostot	valitt_kustannukset
	huonekalut	1220	892	160
	elektroniikka	1130	845	65
	avg	1175	868,5	112,5

Kuva 45: Esimerkkikyselyn 2 lopputulos.

Taulun alimpana rivinä on sarakkeiden keskiarvot (`avg`). Ensimmäisen sarakkeen alin rivi ilmaisee, että myyjän `yksi` tuoteryhmien `huonekalut` ja `elektroniikka` todellisten myyntien keskiarvo on 1175, joka siis on laskun (1220+1130)/2 tulos.

Esimerkkikyselyssä 3 (kuva 46) yhdistyvät perusoperaatioista projektio ja porautuminen.

```
view(proj_drill(vuosipuolisko, tuoteryhma, nuorten_ostot,
vanhojen_ostot), [new_view_dim(nuorten_ostot, ostajaryhma,
[nuoret], todelliset_ostot), new_view_dim(vanhojen_ostot,
ostajaryhma, [vanhat], todelliset_ostot)]).
```

Kuva 46: Esimerkkikysely 3.

Kyselyssä käytetään riippumattomalle muuttujalle `aika` määriteltyä karkeistustasoa `vuosipuolisko` johon kuuluvat arvot `eka_puolisko` ja `toka_puolisko`. `Eka_puolisko` koostuu ensimmäisestä ja toisesta vuosineljänneksestä, `toka_puolisko` taas koostuu kolmannesta ja neljännestä vuosineljänneksestä (ks. kuva 27). Riippumattoman muuttujan `ostajaryhmä` arvoista jätetään huomiotta arvo `keski_ikaiset` (ks. kuva 26). Tämä kysely on esimerkki kyselystä, joka kohdistuu sekä MOLAP –tauluun että karkeistustauluun. Operaation tulos esitellään kuvassa 47.

proj_drill	vuosipuolisko	tuoteryhma	nuorten_ostot	vanhojen_ostot
	eka_puolisko	huonekalut	428	1616
	eka_puolisko	elektroniikka	408	1850
	toka_puolisko	huonekalut	464	1896
	toka_puolisko	elektroniikka	437	2043

Kuva 47: Esimerkkikyselyn 3 lopputulos.

Esimerkkikysely 4 (kuva 48) on muuten sama kuin esimerkkikysely 3 paitsi, että tulostauluun sovelletaan laajennusoperaatioita `row_sums` ja `col_avg`.

```
view(proj_drill_sums(vuosipuolisko, tuoteryhma, nuorten_ostot,
vanhojen_ostot), [new_view_dim(nuorten_ostot, ostajaryhma,
[nuoret], todelliset_ostot), new_view_dim(vanhojen_ostot,
ostajaryhma, [vanhat], todelliset_ostot)]),
add([row_sums(proj_drill_sums), col_avg(proj_drill_sums)]).
```

Kuva 48: Esimerkkikysely 4.

Esimerkki edustaa kyselytyyppiä, joka liittyy samalla kertaa MOLAP –tauluun, karkeistustauluun ja aggregointimuuttujiin.

proj_drill_sums	aika	tuoteryhma	nuorten _ostot	vanhojen _ostot	row_sums
	eka _puolisko	huonekalut	428	1616	2044
	eka _puolisko	elektroniikka	408	1850	2258
	toka _puolisko	huonekalut	464	1896	2360
	toka _puolisko	elektroniikka	437	2043	2480
		avg.	434,25	1851,25	2285,5

Kuva 49: Esimerkkikysleyn 4 tulostaulu.

Esimerkkikyselyssä 5 (kuva 50) yhdistyvät konkatenaatio ja porautuminen.

```
view(conc_drill(vuosipuolisko,  tuoteryhma, nuorten_ostot,
myyja1_myynnit, myyja2_myynnit, myyja3_myynnit, myyja4_myynnit),
[new_view_dim(nuorten_ostot, ostajaryhma, [nuoret],
todelliset_ostot), new_view_dim(myyja1_myynnit, myyja, [yksi],
todellinen_myynti),new_view_dim(myyja2_myynnit, myyja, [kaksi],
todellinen_myynti),new_view_dim(myyja3_myynnit, myyja, [kolme],
todellinen_myynti),new_view_dim(myyja4_myynnit, myyja, [nelja],
todellinen_myynti)]).
```

Kuva 50: Esimerkkikysely 5.

Tässä kyselyssä käyttäjä haluaa yhdistää tietoa myyntitaulusta (`nuorten_ostot`) ja myyjätaulusta (muut sarakkeet) (ks. kuva 14, 15 ja 17). Tässä kyselyssä tarvitaan usean MOLAP –taulun lisäksi karkeistustaulua (ks. kuva 27). Kyselyn tulos annetaan kuvassa 51.

conc_drill	vuosi-puolisko	tuote-ryhma	nuorten_ ostot	myyja1_ myynnit	myyja2_ myynnit	myyja3_ myynnit	myyja4_ myynnit
	eka_ puolisko	huone-kalut	428	550	731	1888	1298
	eka_ puolisko	elektro-niikka	408	600	727	1163	1926
	toka_ puolisko	huone-kalut	464	670	717	2290	1724
	toka_ puolisko	elektro-niikka	437	530	888	1619	2386

Kuva 51: Esimerkkikyselyn 5 tulostaulu.

8.2. OMINAISUUSTAULUJEN YHDISTÄMINEN KYSELYYN

Kehitetty järjestelmä sisältää myös ominaisuustauluja, jotka esitetään kuten relaatiot relaatiotietokannassa. Seuraavaksi tarkastellaan mekanismia, jolla ominaisuustauluja ja MOLAP –tauluja voidaan yhdistää samaan kyselyyn.

Ominaisuustaulujen käsittelyyn käytetään logiikkaohjelmoinnin perusilmauksia, joita voidaan käyttää view ja add –operaatioden ohella.

Esimerkkikyselyssä 6 (kuva 52) käyttäjä haluaa etsiä ominaisuustaulujen avulla niiden kauppojen todelliset ostot, joiden asiakaskunnasta keski-ikäisiä ostajia on alle 200 yksikköä. Riippumattomaan muuttujaan `paikka` liittyvässä ominaisuustaulussa ilmaistaan ostajaryhmien koot kullekin kaupalle.

```
findall(Kauppa, (ostajaryhmien_koot(Kauppa, _, X,_), X<200),
Kauppalista), view(query1(aika, kauppa_myynti),
[new_view_dim(kauppa_myynti, paikka, Kauppalista,
todelliset_ostot)]).
```

Kuva 52: Esimerkkikysely 6.

Kyselyssä käytetään apuna korkealla abstraktiotasolla olevaa `findall(X, G, Solve_list)` –predikaattia, missä `X` on tavoitteessa `G` esiintyvä logiikkaohjelmointimuuttuja ja `Solve_list` koostuu kaikista niista `X`:n arvoista, jotka ovat `G`:n ratkaisuja normaalin Prolog- prosessoinnin mukaisesti [SS94]. Tällä predikaatilla kyselykielessä integroidaan ominaisuustauluissa olevien riippumattomien muuttujien arvot kehitettyjen MOLAP –operaatioiden kanssa. Esimerkkiyselyssä etsitään `ostajaryhmien_koot` –faktoista (ks. kuva 19) sellaiset kaupat (logiikkaohjelmointimuuttuja `Kauppa`), joiden kolmannen argumentin (`X`) arvo eli keski-ikäisten lukumäärä on alle 200 (`X<200`). Jaetun muuttujan `Kauppalista` arvoksi tulee kaikki ko. ehdon täyttävät logikkaohjelmointimuuttuja `Kauppa` arvotukset. Ensimmäinen argumentti `findall` –predikaatissa ilmaisee riippumattomaan MOLAP –muuttujaan `paikka` liittyvän arvon (logiikkaohjelmointimuuttujan `Kauppa` arvo) ja kolmas argumentti ilmaisee keski –ikäisten määrän kyseisellä alueella. View –operaation tulostaulun sarakkeen semanttinen merkitys on siis riippuvainen `Kauppalista` –muuttujan saamasta alustuksesta. Kysely on esimerkki kyselystä, joka yhdistää tietoa MOLAP –taulusta ja ominaisuustaulusta. Esmierkissä `findall` –predikaatti alustaa `Kauppalistan` arvoksi `[kauppa3]`. Tähän kauppaan liittyen annetaan kuvassa 53 oleva kyselyn lopputulos.

query	aika	kauppa_myynti
	ensimmainen	1546
	toinen	1678
	kolmas	1768
	neljas	2342

Kuva 53: Esimerkkikyselyn 6 tuottama vastaus.

Esimerkkikysely 7 (kuva 54) edellyttää navigointia yhden ominaisuustaulun ja MOLAP –taulun välillä.

```
query2(Tuoteryhma, Nimi1, Palkka1, Myynnit1, Nimi2, Palkka2,
Myynnit2):-
view(molap_rel(tuoteryhma, myyja1_myynnit, myyja2_myynnit),
[new_view_dim(myyja1_myynnit, myyja, [yksi],
todellinen_myynti),new_view_dim(myyja2_myynnit, myyja, [kaksi],
todellinen_myynti)]),
molap_rel(Tuoteryhma, Myynnit1, Myynnit2),
myyjien_tiedot(yksi, Nimi1,_,Palkka1),
myyjien_tiedot(kaksi, Nimi2,_,Palkka2).
```

Kuva 54: Esimerkkikysely 7.

Ensimmäisenä kyselyssä annetaan tulosrelaation muoto (yhdistetty termi `query2`) Tulosrelaation ensimmäiseksi sarakkeeksi käyttäjä haluaa riippumattoman muuttujan `tuoteryhmä`, kolmeksi seuraavaksi sarakkeeksi hän haluaa myyjän `yksi` nimen, palkan ja myynnit ko. tuoteryhmästä (ks. kuva 22). Tämän jälkeen hän haluaa vastaavat tiedot myyjään `kaksi` liittyen. View -operaatiossa tutkitaan tuoteryhmittäin myyjien `yksi` ja `kaksi` myyntejä (ks. kuva 17). Kyselyn tuottama tulostaulu visualisoidaan kuvassa 55. Relaatiokyselyssä yhdistetään tietoa view –operaation tulostaulusta sekä myyjiin liittyvästä ominaisuustaulusta.

molap_rel	tuoteryhma	myyja1_myynnit	myyja2_myynnit
	elektroniikka	1130	1615
	huonekalut	1220	1448

Kuva 55: Esimerkkikyselyssä 7 esitetyn view –operaation tulos.

Relaatiokyselyn tulos esitetään kuvassa 56.

query	Tuoteryhma	Nimi1	Palkka1	Myynnit1	Nimi2	Palkka2	Myynnit2
	elektroniikka	arttu	15000	1130	liisa	15000	1615
	huonekalut	arttu	15000	1220	liisa	15000	1448

Kuva 56: Esimerkkikyselyn 7 tulos.

Kysely antaa kaksi vastausta, joista ensimmäinen liittyy tuoteryhmään `huonekalut`, ja toinen tuoteryhmään `elektroniikka`. Tuloksesta ilmenee (kuva 56), että myyjä nimeltä `arttu` saa palkkaa `15000` yksikköä ja että hän on myynyt elektroniikkaa `1130` yksikön edestä ja huonekaluja `1220` yksikön edestä. Tuloksesta ilmenee myös, että myyjä nimeltä liisa saa palkkaa `15000` yksikköä, ja että hän on myynyt elektroniikkaa 1615 yksikön edestä, ja huonekaluja `1448` yksikön edestä.

Seuraavassa esimerkissä kyselyyn lisätään vielä karkeistushierarkioiden käsittely.

```
query2(Tuoteryhma, Nimi1, Palkka1, Myynnit1, Nimi2, Palkka2,
Myynnit2):-
view(molap_rel(kaikki_tuotteet, myyja1_myynnit, myyja2_myynnit),
[new_view_dim(myyja1_myynnit, myyja, [yksi],
todellinen_myynti),new_view_dim(myyja2_myynnit, myyja, [kaksi],
todellinen_myynti)]),
molap_rel(Tuoteryhma, Myynnit1, Myynnit2),
myyjien_tiedot(yksi, Nimi1,_,Palkka1),
myyjien_tiedot(kaksi, Nimi2,_,Palkka2).
```

Kuva 57: Esimerkkikysely 8.

View –operaatiossa käyttäjä haluaa tutkia jälleen myyjien `yksi` ja `kaksi` myyntejä. Nämä tiedot ryhmitellään riippumattoman muuttujan `tuoteryhma` karkeistustason `kaikki_tuotteet` (ks. kuva 24) mukaan. View –operaation tulos esitellään kuvassa 58 visualisoituna.

molap_rel	kaikki_tuotteet	myyja1_myynnit	myyja2_myynnit
	kaikki_tuotteet	2350	3063

Kuva 58: Esimerkkikyselyn 8 view –operaation tulos visualisoituna.

Kyselyssä suoritettava relaatiokysely on vastaava kuin kuvassa 54 esitetty. Relaatiokyselyn tulos esitetään kuvassa 59.

query	Tuoteryhma	Nimi1	Palkka1	Myynnit1	Nimi2	Palkka2	Myynnit2
	kaikki_tuotteet	arttu	15000	2350	liisa	15000	3063

Kuva 59: Esimerkkikyselyn 8 tulos

Seuraavassa (kuva 60) kyselyssä mukaan lisätään vielä aggregoidut muuttujat, jolloin kyselyssä käytetään apuna MOLAP –taulua, karkeistushierarkioita, ominaisuustaulua ja aggregointimuuttujia. Kysely siis kohdistuu kaikkiin järjestelmässä kuvattuihin osiin.

```
query2(Tuoteryhma, Nimi1, Palkka1, Myynnit1, Nimi2, Palkka2,
Myynnit2, Yhteensa):-
view(molap_rel(kaikki_tuotteet, myyja1_myynnit, myyja2_myynnit),
[new_view_dim(myyja1_myynnit, myyja, [yksi],
todellinen_myynti),new_view_dim(myyja2_myynnit, myyja, [kaksi],
todellinen_myynti)]),
add([row_sums(molap_rel)]),
molap_rel(Tuoteryhma, Myynnit1, Myynnit2, Yhteensa),
myyjien_tiedot(yksi, Nimi1,_,Palkka1),
myyjien_tiedot(kaksi, Nimi2,_,Palkka2).
```

Kuva 60: Esimerkkikysely 9.

Kysely on samankaltainen kuin edellinen kysely (kuva 57). Erona on MOLAP –taulun laajennusoperaation `row_sums` soveltaminen tulostauluun ennen relaatiokyselyn suorittamista, joka lisää tulostauluun yhden sarakkeen. Tämä tieto on lisätty myös tulosrelaatioon (`Yhteensä`). Tulostaulu esitetään kuvassa 61.

molap_rel	kaikki_tuotteet	myyja1_myynnit	myyja2_myynnit	row_sums
	kaikki_tuotteet	2350	3063	5413

Kuva 61: Esimerkkikyselyn 9 view –operaation tulos.

Tulostauluun on lisättu sarake `row_sums`, joka ilmaisee myyjien `yksi` ja `kaksi` yhteismyynnin tuoteryhmittäin. Kyselyyn liittyvän relaatiokyselyn tulos esitetään kuvassa 62.

query	Tuoteryhma	Nimi1	Palkka1	Myynnit1	Nimi2	Palkka2	Myynnit2	Yhteensa
	kaikki_tuotteet	arttu	15000	2350	liisa	15000	3063	5413

Kuva 62: Esimerkkikyselyn 9 tulos.

8.3. KETJUTETUT OPERAATIOT

Seuraavaksi esitellään jatkokyselyn suorittamista perustuen view –operaatiolla tuotettuun tulostauluun. Kuvassa 63 esitellään peruskysely, joka luo `first` –nimisen tulostaulun. Esimerkkikysely perustuu myyntitauluun, johon liittyvät tiedot esitetään kuvassa 15.

```
view(first(paikka, aika, nuorten_ostot, keski_ikaisten_ostot,
vanhojen_ostot), [new_view_dim(nuorten_ostot, ostajaryhma,
[nuoret], todelliset_ostot),new_view_dim(keski_ikaisten_ostot,
ostajaryhma, [keski_ikaiset],
todelliset_ostot),new_view_dim(vanhojen_ostot, ostajaryhma,
[vanhat], todelliset_ostot)]).
```

Kuva 63: Esimerkkikysely 10.

Kyselyssä (kuva 63) käyttäjä haluaa ostajaryhmien todelliset ostot ryhmiteltynä paikan ja ajan mukaan. Operaatiossa suoritetaan projektio, koska riippuvista muuttujista otetaan huomioon vain `todelliset_ostot` ja riippumaton muuttuja `tuoteryhmä` on jätetty pois (ks. kuva 15). Tämä kysely on esimerkki kyselystä joka kohdistuu vain yhteen MOLAP –tauluun. Kyselyn tulos esitetään kuvassa 64 visualisoituna.

first	paikka	aika	nuorten_ostot	keski_ikaisten_ostot	vanhojen_ostot
	kauppa1	ensimmainen	151	850	370
	kauppa1	toinen	137	740	360
	kauppa1	kolmas	164	930	280
	kauppa1	neljas	146	965	280
	kauppa2	ensimmainen	140	814	577
	kauppa2	toinen	127	834	559
	kauppa2	kolmas	140	835	706
	kauppa2	neljas	170	1052	1006
	kauppa3	ensimmainen	138	714	694
	kauppa3	toinen	143	629	906
	kauppa3	kolmas	111	964	693
	kauppa3	neljas	170	1238	934

Kuva 64: Esimerkkikyselyn 10 tulos.

Kyselyn jälkeen voidaan luoda näkemystauluun perustuvia kyselyitä. `Nuorten_ostot`, `keski_ikaisten_ostot` ja `vanhojen_ostot` ovat nyt tulostaulussa olevien riippuvien muuttujien nimiä. Kuvassa 64 esitellyn tulostaulun pohjalta tehdään kuvassa 65 jatkokysely.

```
view(second(aika, n_ita, n_etela, k_ita, k_etela),
[new_view_dim(n_ita, paikka, [kauppa1], nuorten_ostot),
new_view_dim(n_etela, paikka, [kauppa2, kauppa3],
nuorten_ostot), new_view_dim(k_ita, paikka, [kauppa1],
keski_ikaisten_ostot), new_view_dim(k_etela, paikka, [kauppa2,
kauppa3], keski_ikaisten_ostot)]).
```

Kuva 65: Esimerkkikysely 11.

View –operaatio tunnistaa first –taulun olemassaolon, vaikka siihen ei eksplisiittisesti viitata ja suorittaa kyselyn sen pohjalta. Kysely sisältää projektion. Riippuvista muuttujista jätetään huomiotta `vanhojen_ostot`. Nyt sarakkeilla esitetään riippumaton muuttuja `aika` ja muita sarakkeita määrittelee riippumattoman muuttujan `paikka` arvot. Jatkokyselyn tulos esitetään kuvassa 66 visualisoituna.

second	aika	n_ita	n_etela	k_ita	k_etela
	ensimmainen	151	278	850	1528
	toinen	137	270	740	1463
	kolmas	164	251	930	1799
	neljas	146	340	965	2290

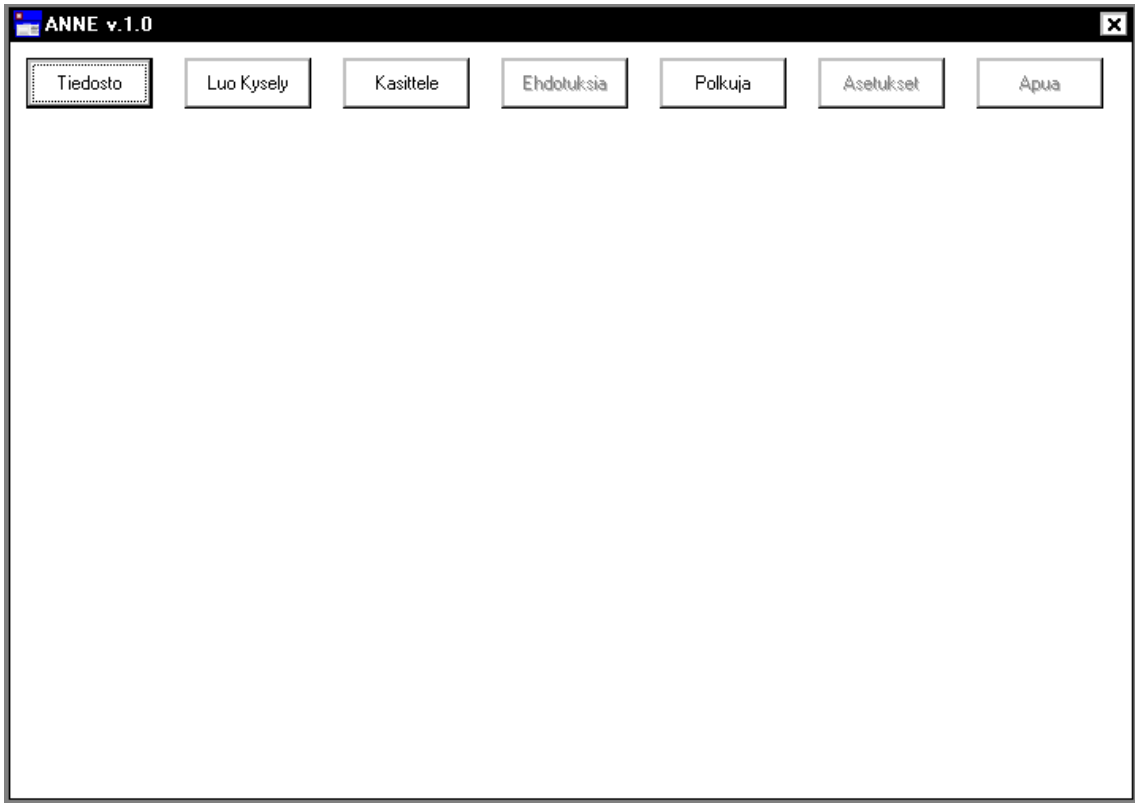
Kuva 66: Esimerkkikyselyn 11 tulos.

9. Graafinen käyttöliittymä

Seuraavaksi esitellään prototyyppiin kehitetty graafinen käyttöliittymä Anne. Graafisen käyttöliittymän tavoitteena on tehdä kyselyjen tekemisestä vielä helpompaa tekstuaaliseen käyttöliittymään verrattuna. Siinä loppukäyttäjän ei enää tarvitse muistaa riippuvien tai riippumattomien muuttujien nimiä eikä view – operaation argumenttien järjestystä tai esittämistapoja.

9.1. GRAAFISEN KÄYTTÖLIITTYMÄN NÄYTÖT

Graafisen käyttöliittymän suunnittelussa on pyritty yksinkertaiseen ja helppokäyttöiseen visuaaliseen ilmeeseen. Käyttöliittymän alkunäyttö esitellään kuvassa 67.

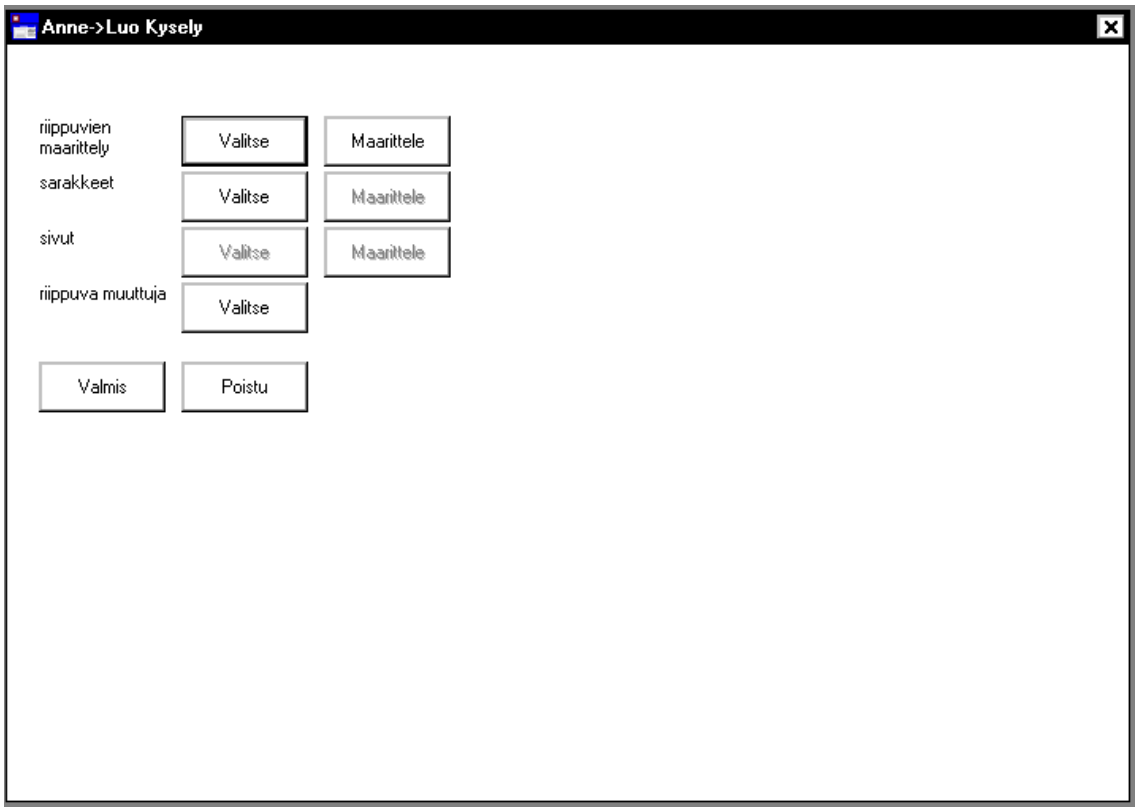


Kuva 67: Alkunäyttö graafisesta käyttöliittymästä.

Graafisen käyttöliittymän nykyisessä l. ensimmäisessä versiossa on jätetty osa käyttöä tukevasta toiminnallisuudesta (painikkeet ’Asetukset’ ja ’Apua’) ja osa tekoälytoiminnallisuudesta toteuttamatta (painike ’Ehdotuksia’). Aiemmin esitetyllä tekstuaalisella käyttöliittymällä saadaan myös joitakin sellaisia kyselyyn vaikuttavia tekijöitä määriteltyä, joita graafisella käyttöliittymällä ei toistaiseksi kyetä ilmaisemaan. Esimerkiksi nykyinen graafinen käyttöliittymä ei tue

ominaisuustaulujen käsittelyä. Kehitetyn graafisen käyttöliittymän päätarkoitus on osoittaa, että tekstuaalisen käyttöliittymän piirteet ovat ilmeisellä tavalla visualisoitavissa.

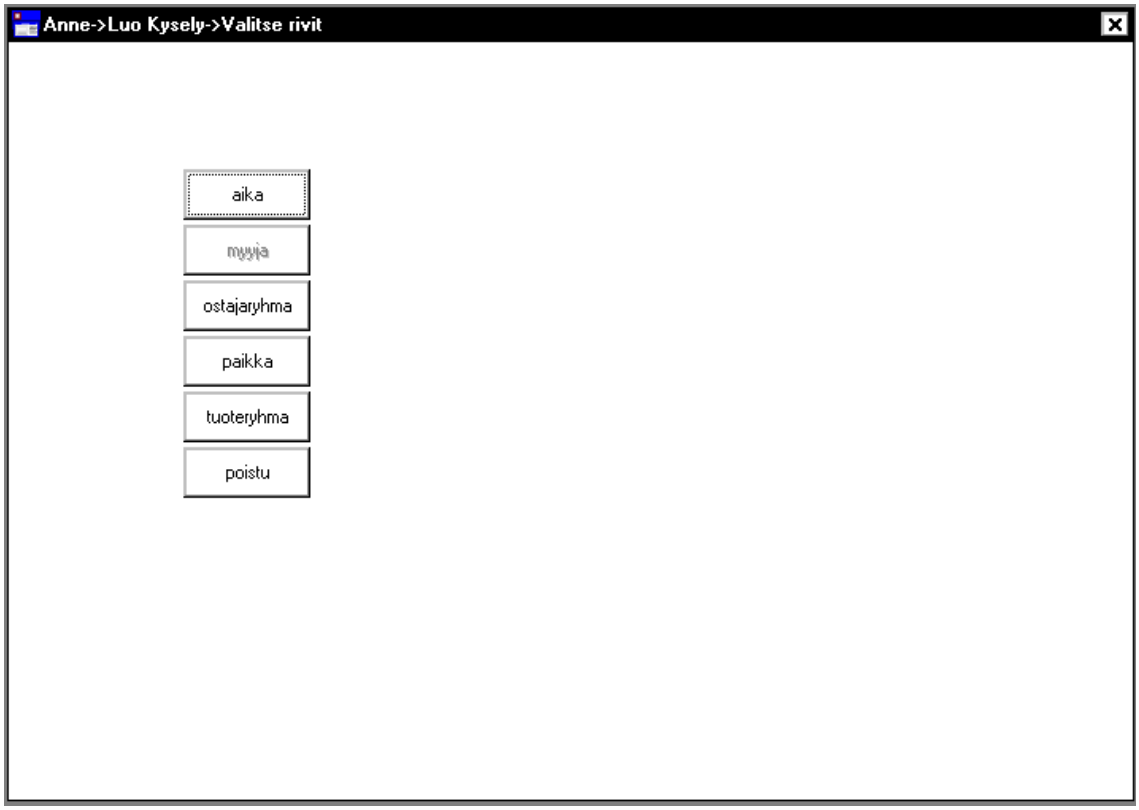
Painike ’Tiedosto’ avaa valikon, jonka avulla voidaan tallentaa tiedostoja ja avata tallennettuja tiedostoja, tulostaa MOLAP -näkymä tai poistua ohjelmasta. Painike ’Luo Kysely’ avaa kyselynäytön (kuva 68).



Kuva 68: Graafisen käyttöliittymän kyselynäyttö.

Kyselynäytön avulla käyttäjä määrittelee itse asiassa view –operaation tarvitseman informaation siten, että hänen ei tarvitse olla tietoinen argumenttien muodosta tai järjestyksestä. Järjestelmä muistaa käyttäjän istunnon aikana tekemät valinnat. Riippumattomien muuttujien määrittelyyn (’riippuvien määrittely’) ja sarakkeisiin (’sarakkeet’) liittyvät painikkeet ’Valitse’ avaavat riippumattomien muuttujien valintanäytön (kuva 69). Painikkeen ’Valmis’ avulla käyttäjä ilmaisee, että kaikki valinnat on tehty. Sivujen määrittelyä ei tähän versioon toteutettu. Sivujen määrittelemisellä ymmärretään mekanismia, jonka perusteella luodaan useita näkemystauluja, joista kukin liittyy johonkin riippumattoman muuttujan arvoon. Tällöin ko. riippumaton muuttuja määrittelee sivut. Käyttöliittymä on toteutettu

siten, että käyttäjän valinnoista muodostetaan edellä esitelty tekstuaalinen view – operaatio, joka ajetaan.



Kuva 69: Graafisen käyttöliittymän riippumattomien muuttujien valintanäyttö.

Riippumattomien muuttujien valintanyttö antaa käyttäjälle mahdollisuuden valita kyselyyn tulevat riippumattomat muuttujat ilman että hänen tarvitsee muistaa niiden nimiä. Näytön kaikista painikkeista (paitsi 'poistu' painikkeesta) avautuu karkeusasteen valintaikkuna, jossa kuvataan kyseiseen riippumattomaan muuttujaan liittyvän karkeistushierarkian kaaviotaso, josta käyttäjä valitsee haluamansa karkeistustason. Kyselyn antamassa tulostaulussa kyseinen riippumaton muuttuja esitetään valitulla karkeistustasolla.

9.2. KYSELYJEN TEKEMINEN GRAAFISELLÄ KÄYTTÖLIITTYMÄLLÄ

Seuraavaksi esitellään esimerkkikyselyn tekeminen kehitetyllä graafisella käyttöliittymällä. Kyselynäytössä valitaan riippumattonta muuttuja määrittelevä riippuva muuttuja, sarakkeet ja riippuva muuttuja. Kyselyssään käyttäjä voi valita muuttujat missä tahansa järjestyksessä. Esimerkkikyselyssä käyttäjä haluaa riippuvan muuttujan todelliset ostot eriteltynä kaupoittain ja ryhmiteltynä vuosineljänneksittäin. Esimerkkikyselyssä hän valitsee ensin sarakkeiden

riippumattoman muuttujan. Tämä tapahtuu klikkaamalla ohjaustekstin ’sarakkeet’ vieressä olevaa ’Valitse’ painiketta. Seurauksena aukeaa kuvassa 69 esitetty riippumattoman muuttujan valintanäyttö, joka ilmaisee sovellutuksen kaikki riippumattomat muuttujat.

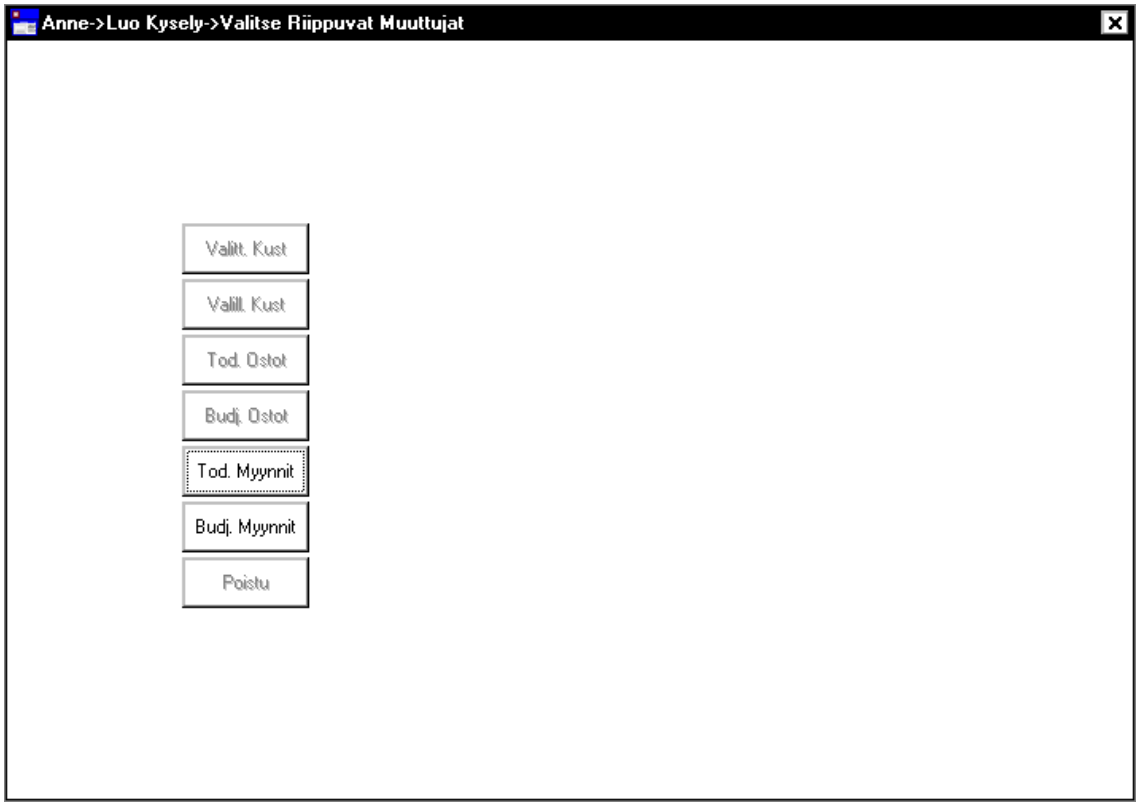
Nyt käyttäjä voi valita riippumattoman muuttujan sarakkeille tarvitsematta muistaa mitä riippumattomia muuttujia sovellus sisältää. Esimerkkikyselyssä käyttäjä valitsee riippumattoman muuttujan `aika`. Valinnan seurauksena aukeaa riippumattoman muuttujan karkeistusvaihtoehdot. Tämä näyttö kuvataan kuvassa 70.

Kuva 70: Graafisen käyttöliittymän riippumattoman muuttujan aika karkeistusastenäyttö

Tämän näytön yhteydessä käyttäjä valitsee haluamansa karkeistusasteen riippumattomalle muuttujalle `aika`. Esimerkkikyselyssä käyttäjä valitsee karkeistustason ’Neljannes’. Valinnan jälkeen palataan kuvassa 68 esitettyyn näyttöön, jolloin ohjaustekstin ’sarakkeet’ vieressä oleva ’Valitse’ -painike ei kuitenkaan ole enää valittavissa. Käyttäjän tekemä valinta `aika` neljännesvuosien tarkkuudella saa aikaan `aika` sarakkeen muodostamisen kyselyn lopputulokseen. Kyselyn tekemistä voidaan jatkaa valitsemalla ohjaustekstin ’riippuvien määrittely’ vieressä oleva ’Valitse’ –painike, jolloin valinta tapahtuu samalla tavalla kuin

sarakkeiden tapauksessa (kuvat 69-70). Esimerkkikyselyssä käyttäjä valitsee riippuvia muuttujia määrittteleväksi riippumattomaksi muuttujaksi paikan, ja sille karkeistusasteeksi kaupan (karkeistusasteina ovat maa, alue ja kauppa).

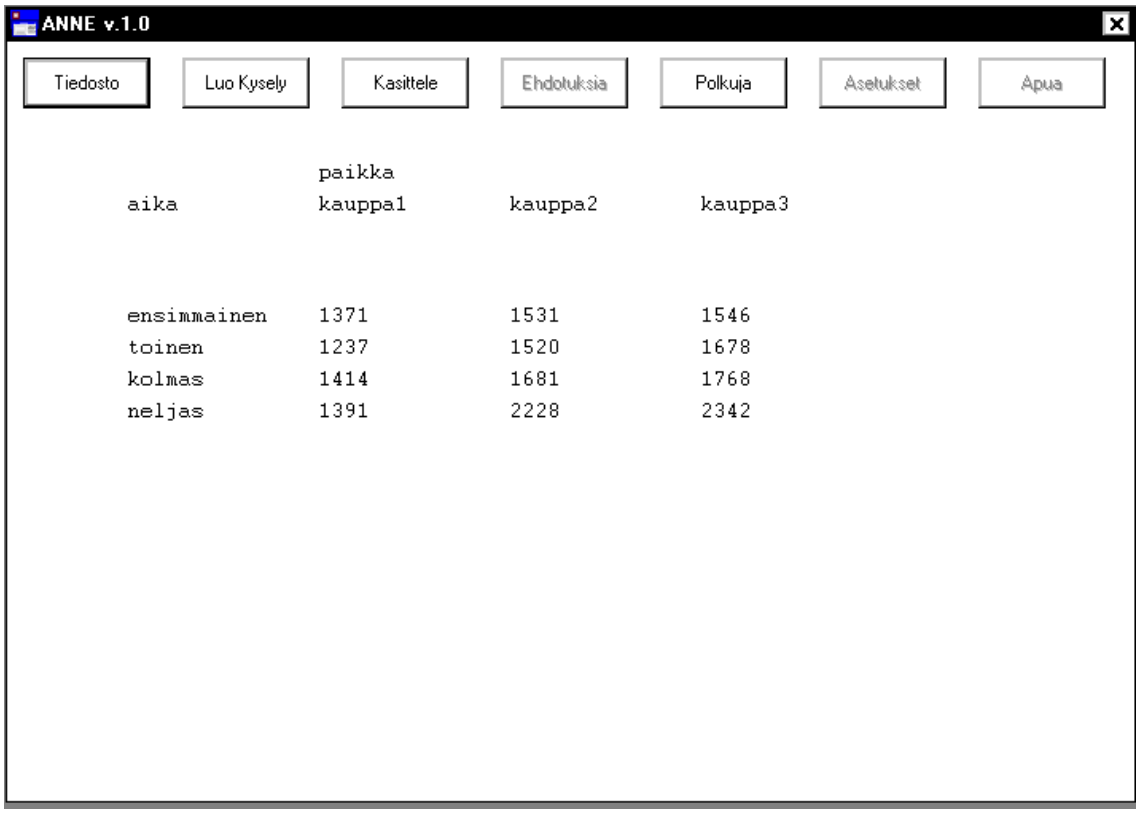
Nyt esimerkkikyselyn määrittelystä puuttuu enää riippuvan muuttujan valitseminen. Tämä tapahtuu kuvassa 68 esitellyn näytön ohjaustekstin ’riippuva muuttuja’ vieressä olevan ’Valitse’ –painikkeen avulla. Valinnasta seuraava näkymä esitellään kuvassa 71.



Kuva 71: Graafisen käyttöliittymän riippuvan muuttujan valintanäyttö.

Näytössä listataan kaikki sovellutuksen riippuvat muuttujat. Esimerkin tilanteessa vain painikkeet ’Tod. Ostot’ ja ’Budj. Ostot’ ovat valittavissa. Tämä johtuu käyttäjän valitsemista riippumattomista muuttujista. Kuten kuvasta 14 huomataan, riippumattomat muuttujat AIKA ja PAIKKA määrittelevät kaksi riippuvaa muuttujaa: todelliset ostot ja budjetoidut ostot. Esimerkkitilanteessa käyttäjä valitsee painikkeen ’Tod. Ostot’. Valinnan jälkeen palataan kuvassa 68 esiteltyyn näyttöön, jossa valittavana on enää kaksi painiketta: ’Valmis’ ja ’Poistu’. Niistä ensimmäinen suorittaa valitun kyselyn, ja jälkimmäinen poistaa käyttäjän tekemät valinnat muistista. Kummassakin tapauksessa palataan kuvassa 67 esitettyyn näyttöön. Esimerkkikyselyssä käyttäjä valitsee painikkeen ’Valmis’. Nyt graafinen

käyttöliittymä kokoaa käyttäjän valinnoista view –operaation, jonka se lähettää tekstuaaliselle käyttöliittymälle. Tämän jälkeen graafinen käyttöliittymä visualisoi tulostaulun. Kyselyn tulos esitetään kuvassa 72.



Kuva 72: Graafisen käyttöliittymän visualisoima esimerkkikyselyn tulos.

Tulosta voitaisiin käsitellä edelleen painikkeen ’Käsittele’ avulla. Sillä voitaisiin lisätä tauluun sekä rivi- että sarakesummat tai rivien ja sarakkeiden keskiarvot. Näytön painike ’Käsittele’ toteutetaan siis aiemmin esitellyn tekstuaalisen add –operaation avulla.

Kuvassa 68 esitellyssä kyselynäkymässä on myös sarakkeisiin liittyvä ’Määrittele’ -painike. Tämän painikkeen avulla voidaan määritellä jokainen tulostaulun sarake erikseen. Sen avulla siis voidaan ilmaista pyöristyksiä sarakkeille.

9.3. GRAAFISEN KÄYTTÖLIITTYMÄN TEKOÄLY

Graafiseen käyttöliittymään on luotu myös yksinkertainen tekoäly, joka helpottaa usein toistuvien kyselyjen tekemistä. Toisin sanoen kyselyjärjestelmä analysoi tehtyjä kyselyjä ja päättelee usein toistuvat kyselyt, joiden pohjalta se päivittää painikkeen ’Polkuja’ takana olevaa valikkoa.

10. Yhteenveto

Tutkielmassa määriteltiin käsitteistö, jonka avulla mallinnetaan MOLAP –tietokuutioita systemaattisesti ja yleisesti. Esittämällä kaaviotaso ja ilmentymätaso erikseen ja luomalla täsmällinen mekanismi tasojen liittämiseksi toisiinsa mahdollistetaan yleisten OLAP –operaatioiden kehittäminen.

Tutkielmassa suunniteltiin ja toteutettiin ilmaisuvoimainen kyselykieli, jonka avulla käyttäjä voi tehdä kyselyjä MOLAP –tietokuutioon. Kieli sisältää kaksi voimakasta operaatiota: view –operaation ja add –operaation. Operaatiot ovat korkealla abstraktiotasolla ja niiden käyttäminen on intuitiivista loppukäyttäjälle. View –operaation avulla kuvataan tulostaulun muoto ja ne tekijät, jotka määrittelevät riippuvien muuttujien arvoja. View –operaatiolla voidaan esittää mikä tahansa perinteisten OLAP –järjestelmien mahdollistamien operaatioiden kombinaatio. Kehitetty view –operaatio kykenee automaattiseen ja loppukäyttäjälle näkymätömään navigointiin MOLAP –taulujen välillä. Soveltaessaan view –operaatiota käyttäjä deklaratiivisesti määrittelee sekä halutun tulostaulun rakenteen että tulostaulun sarakkeiden johtamisen sovellutuksen tunnistamista riippuvista ja riippumattomista muuttujista. Add –operaatio tuottaa MOLAP –tauluun aggregointitietojen aggregointeja.

View –operaatio navigoi ilman käyttäjän kontrollia MOLAP –taulujen, aputietotaulujen ja näkemystaulujen välillä. Vaikka navigointia ei tarvitse määritellä, on käyttäjän kuitenkin tiedettävä, miten eri MOLAP –tauluissa olevat riippumattomat ja riippuvat muuttujat liittyvät toisiinsa. Add –operaatioa sovelletaan usein view –operaation yhteydessä, vaikka se on täysin itsenäinen operaatio. View ja Add –operaatioiden lisäksi kyselykieli koostuu logiikkaohjelmoinnin peruskäsitteistä, joita käytetään kehitettyjen operaatioiden ja ominaisuustauluissa olevien relationaalisesti organisoitujen tietojen integrointiin.

Prototyyppiin toteutettu graafinen käyttöliittymä vapauttaa loppukäyttäjän muistamasta sovellukseen määriteltyjen muuttujien nimiä ja niiden sijainteja MOLAP –tauluissa. Graafinen käyttöliittymä käyttää myös yksinkertaista tekoälyä auttamaan loppukäyttäjän usein suorittamia kyselyjä luoden painikkeita, joiden avulla usein toistuvat kyselyt voidaan suorittaa nopeasti.

Tutkielmassa kehitetty tekstuaalinen ja graafinen kyselykieli sekä MOLAP – kuutioiden organisointi toteutettiin LPA Win –Prolog –ohjelmistolla.

11. Lähteet

[AAD+96] S. Agrawal, R. Agrawal, P.M. Deshpande, A. Gupta, J.F. Naughton, R. Ramakrishnan, S. Sarawagi. On the Computation of Multidimensional Aggregates. *Proceedings of the 22nd VLDB Conference 1996.* ss 506-521.

[AGG+98] R. Agrawal, J. Gerke, D. Gunopulos, P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. *SIGMOD 1998.* ss 94-105.

[AGS97] R. Agraval, A. Gupta, S. Sarawagi. Modeling Multidimensional Databases. *Proceedings of the 13th Int'l Conference on Data Engineering. Birmingham, U.K., April 1997.* http://www.almaden.ibm.com/cs/quest. Luettu 4.12.2000.

[BKK96] S. Berchtold, D.A. Keim, H-P. Kriegel. The X-tree: An Index Structure for High-Dimensional Data. *Proceedings of the 22nd VLDB Conference 1996.* ss 28-39.

[BPT97] E. Baralis, S. Paraboschi, E. Teniente. Materialized View Selection in a Multidimensional Database. *Proceedings of the 23rd VLDB Conference 1997.* ss 156-165.

[BS97] D. Barbará, M. Sullivan. Quasi-Cubes: Exploiting approximations in multidimensional databases. *SIGMOD Records, Vol. 26, No. 3, September 1997.* ss 12-17.

[CAA+99] S. Ceffner, D. Agrawal, A. El Abbadi, T. Smith. Relative Prefix Sums: An Efficient Approach for Querying Dynamic OLAP Data Cubes. *15th International Conference on Data Engineering 1999.* ss 328-335.

[CC93] E.F.Codd, S.B.Codd, C.T.Salley. Providing OLAP to User-Analysts: An IT Mandate. In *E.F.Codd & Accosiates, available at www.hyperion.com*, 1993. Luettu 4.12.2000.

[Dat00] C. Date. *An Introduction to Database Systems*, 7th edititon. Addison-Wesley, Reading MA, 2000.

[DRS+98] P.M. Deshpande, K. Ramasamy, A. Shukla, J.F. Naughton. Caching Multidimensional Queries Using Chunks. *SIGMOD 1998.* ss 259-270.

[Dyr96] C. Dyreson. Information Retrieval from an Incomplete Data Cube. *Proceedings of the 22nd VLDB Conference 1996* ss. 532-543.

[FB99] P. Furtado, P. Baumann. Storage of Multidimensional Arrays Based on Arbitrary Tiling. *15th International Conference on Data Engineering 1999.* ss 480-489.

[GBL+95] J. Gray, A. Bosworth, A. Layman, H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *12th International Conference on Data Engineering 1996*, ss 152–159.

[GL97] M. Gyssens, L.V.S Lakshmanan. A Foundation for Multi-Dimensional Databases. *Preceedings of VLDB Conference 1997*. ss 106-115.

[GL98] F. Gingras, L.V.S.Laksmanan. nD-SQL: A Multidimensional Language for Interoperability and OLAP. *Proceedings of VLDB Conference 1998.* ss 134-145.

[GPS00] J. Gu, T. B. Pedersen, A. Shoshani. OLAP++: Powerful and Easy-to-Use Federations of OLAP and Object Databases. *Proceedings of VLDB Conference 2000.* ss 599-602.

[HMV99] C.A. Hurtado, A.O. Mendelzon, A.A. Vaisman. Maintaining Data Cubes Under Dimension Updates. *15th International Conference on Data Engineering 1999.* ss 346-357.

[Hyp] Hyperion Essbase Technologies. http://www.hyperion.com/eo.cfm. Luettu 4.12.2000.

[Inf a] Informix. Informix Metacube 4.2 http://www.informix.com/informix/products/tools/metacube/datasheet.htm. Luettu 4.12.2000.

[Inf b] Informix. Data Warehouse Servers: the Evolution of Integrated Relational OLAP. http://www.informix.com/informix/whitepapers/rolapwp.pdf. Luettu 13.1.2001

[Kos98] M. Koskinen. *Tiedon reaaliaikainen analysointi*. Pro-Gradu-tutkielma, Tampereen Yliopisto, Tietojenkäsittelyopin laitos, 1998.

[Liu99] M. Liu. Deductive Database Languages: Problems and Solutions. ACM Computing Surveys, Vol. 31, No. 1, March 1999. ss 27 – 62.

[Mel94] J. Melson (editor). ISO-ANSI, Working Draft Database Language SQL/ Foundation (SQL3), Part 2. August, 1994); American National Standards Institute. http://epoch.cs.berkeley.edu:8000/sequoia/schema/ STANDARDS/SQL3/sql3part2.txt

[Mic] MicroStartegy. The Case for Relational OLAP, MicroStrategy, Inc. http://www.microstrategy.com/Publications/WhitePapers/Case4Rolap/execsumm.htm, Luettu 28.2.2001.

[MLC00] M.C.McCabe, J.Lee, A.Chowdury, D.Grossman, O.Frieder. On the Design and Evaluation of a Multi-dimensional Approach to Information Retrieval. *SIGIR 2000 7/00 Athens, Greece.* ss 363-365.

[MQM97] I.S. Mumick, D. Quass, B.S. Mumick. Maintenance of Data Cubes and Summary Tables in a Warehouse. *SIGMOD 1997.* ss 100-111.

[NJ91] T. Niemi, K. Järvelin. Prolog-Based Meta-Rules for Relational Database Representation and Manipulation. *IEEE Transactions on Software Engineering, vol 17, No. 8, August 1991.* ss 762-788.

[NNT00] T. Niemi, J. Nummenmaa, P. Thanisch. Functional Dependencies in Controlling Sparsity of OLAP Cubes. *Proceedings of the 2nd International Conference on Data Warehousing an Knowlwdge Discovery, London , September 2000*. ss 199-209.

[PJ99] T.B.Pedersen, C.S.Jensen. Multidimensional Data Modeling for Complex Data. *Proceedings of the 15th International Conference on Data Engineering, Sydney, Austrialia. IEEE Computer Society 1999.* ss 336-345.

[RS97] K.A. Ross, D. Srivastava. Fast Computation of Sparse Datacubes. *Proceedings of the 23rd VLDB Conference 1997*. ss 116-125.

[SS94] L. Sterling, E. Shapiro. *The Art of Prolog: Advanced Programming Techniques*, The MIT Press, second edition. UK, 1994.

[Tam98] Y.J.Tam. *Datacube: Its Implementation and Application in OLAP Mining*. Thesis for the Degree of Master of Science. Simon Fraser University, 1998. Available at http://tiger.ees.kyushu-u.ac.jp/~hu/reference.html. Luettu 4.12.2000.

[Tho97] Erik Thomsen. *OLAP Solutions: Building Multidimensional Information System*. Wiley Computer Publishing. Chichester UK, 1997.

[Ull88] J. D. Ullman. *Principles of Database and Knowledge –base Systems, Volume I.* Computer Science Press, 1988.

[ZDN97] Y. Zhao, P.M. Deshpande, J.F. Naughton. An Array Based Algorithm for Simultaneous Multidimensional Aggregates. *SIGMOD 1997*. ss 159-169.