TIETOKANTOJEN INTEGROINTIMAHDOLLISUUKSIA

Peter Ylämö

27.04.2004

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

TIIVISTELMÄ

Informaatio on avainresurssi organisaatioiden päivittäisissä operaatioissa. Lisääntyneen tiedontarpeen vuoksi tietojen saatavuus eri lähteistä on tärkeää. Tiedon hankinta useista lähteistä on johtanut suureen määrään erilaisia sovelluksia, joiden tarkoitus on mahdollistaa tiedonhaku erityyppisistä tietokannoista. Järjestelmien kirjo edellyttää useiden erilaisten tiedonhakumenetelmien ja käyttöliittymien omaksumista. Useiden hakumenetelmien opettelu ja käyttö on kallista.

Useiden tiedon varastointipaikkojen välinen vuorovaikutus helpottaa tiedonhankintaa. Tietokantojen integrointi ja monitietokantajärjestelmien luonti mahdollistavat tiedonhaun yhden käyttöliittymän kautta useista tietokannoista. Tietovarastojen luonti on toinen mahdollisuus nopeuttaa tiedonhankintaa. Tietovarastointi edellyttää eriasteista lähdetiedon yhdistämistä. Tiedon käsitteleminen useiden lähteiden kesken on mahdollista välittäjien avulla. Välittäjät sekä välittävät että käsittelevät tietoa. Kääreet toimivat välittäjien ja tietolähteiden välissä ja välittäjien apuna tiedon muuntamisessa. Erityisesti WWW-sovelluksissa datan integroinnissa ja siirrossa käytettävä XML-kieli mahdollistaa monipuolisuutensa vuoksi informaation käsittelyn ja varastoinnin ilman suuria investointeja.

Tässä tutkielmassa on tarkoitus selvittää kirjallisuuden perusteella mitä tarkoitetaan tietokantojen yhdistämisellä sekä tietovarastoinnilla ja välittäjillä datan integroinnissa. Lisäksi tarkastellaan kääreiden osuutta informaation hankintaan ja datan muuntamiseen sekä XML-kielen käyttöä lähdetiedon yhdistämisessä.

ACM-luokitus: (ACM Computing Classification System, 1998 version) H.2

Avainsanat: tietokantojen integrointi, tietovarastointi, välittäjä, kääre, XML, lähdetiedon yhdistäminen

SISÄLTÖ

1 JOHDANTO 1
2 YHDISTETTY TIETOKANTAJÄRJESTELMÄ 4
2.1 Yhdistämisen periaate 4
2.2 Yhdistetty tietokannan hallintajärjestelmä 6
2.2.1 Ominaisuudet 7
2.2.2 Arkkitehtuuri 8
3 TIETOVARASTOINTI 14
3.1 Tietovaraston periaate 14
3.2 Arkkitehtuuri 17
3.2.1 Moniulotteisuus 17
3.2.2 Indeksointi 20
3.2.3 Paikallisvarastot 21
3.3 Tietovaraston rakentaminen 21
3.3.1 Datalle asetetut vaatimukset 22
3.3.2 Datan puhdistaminen 23
3.3.3 Datan lataaminen 24
3.4 Toiminnallisuus 25
4 VÄLITTÄJÄT 27
4.1 Periaate 27
4.2 Tietolähteiden mallinnus 30
4.3 Ontologia 32
4.4 Kääreet 34
4.4.1 Mallinteet 36
4.4.2 Suodattimet 37
5 XML 39
5.1 XML-dokumentti 39
5.1.1 Datakeskeinen XML-dokumentti 41
5.1.2 Dokumenttikeskeinen XML-dokumentti 43
5.2 Datan integrointi ja XML 44
6 YHTEENVETO 49
VIITELUETTELO 52

1 JOHDANTO

Tietokantojen integrointi on tietojen yhdistämiseksi ja tiedon saatavuuden parantamiseksi kehitetty menetelmä. Parentin ja Spaccapietran (1998) mukaan tietolähteiden yhteensovittaminen prosessina sisältää joukon syötetietona toimivia tietokantoja ja tuotoksena on yksi yhdistetty kaavojen kuvaus, johon liittyy kuvaus yhdistetyn tietojoukon tuki-informaatiosta. Yhdistettäessä tietokantoja useiden tietokantojen ja niiden hallintajärjestelmien käsittäväksi joukoksi saadaan aikaan joukko hajautettuja, paikallisia ja autonomisia järjestelmiä, joiden toimintaa kontrolloi yhdistetty tietokannan hallintajärjestelmä. Ghidinin ja Serafinin (1998) mukaan hajautettu tarkoittaa sitä, että yhdistettyyn tietokantajärjestelmään kuuluvat tietokannat sijaitsevat eri järjestelmissä. Autonomia tarkoittaa, että yhdistettyyn monitietokantajärjestelmään kuuluvilla yksittäisillä tietokantajärjestelmillä on jonkin asteinen itsenäisyys suorittaa tiettyjä toimenpiteitä ilman, että yhdistetty tietokannan hallintajärjestelmä on näistä toimenpiteistä tietoinen. Järjestelmän tietokantoihin on mahdollista päästä suorittamaan operaatioita sekä paikallistasolla että globaalisti yhdistetyn tietokannan hallintajärjestelmän kautta.

Theodoratosin ja Sellisin (1999) mukaan tietovarasto on joukko materialisoituja näkemiä, jotka on määritelty toisistaan erillään olevista relaatioista. Tietovarasto suunnitellaan tiedonhakua varten. Tietovarastoon tallennetaan kiinnostuksen kohteena olevan kohdealueen tietoa. Tietovaraston tarkoitus on tarjota hyvällä vasteajalla oikeita vastauksia käyttäjien kyselyihin ja tuottaa koostettua tietoa. Tietokantajärjestelmät, kuten tietovarastot, sisältävät Kedadin ja Métaisin (1999) mukaan yhden tai useamman heterogeenisen tietolähteen. Tietojen hakeminen useasta tietokannasta voi aiheuttaa ongelmia varsinkin semantiikan suhteen, koska samaa tarkoittava informaatio saattaa olla tallennettu eri muodossa ja erilaista rakennetta käyttäen. Tietorakenteiden ja semantiikan eroavuuksien takia tietoa joudutaan sovittamaan yhteen, puhdistamaan ja yhdenmukaistamaan ennen tiedon tallennusta tietovarastoon. Tiedon yhdistämistä varten on olemassa menettelytapoja, joiden avulla tietoa haetaan, muunnetaan oikeaan muotoon ja esitetään käyttäjälle.

Wiederholdin (1997) mukaan välittäjät tukevat virtuaalista näkemää tai kokoelmaa näkemistä, jotka integroivat useita tietolähteitä samaan tapaan kuin materialisoidut relaatiot yhdistävät tietovaraston tietoa. Välittäjien ja tietovarastojen mekaniikka on kuitenkin melko erilaista, sillä välittäjät eivät varastoi tietoa. Tietojärjestelmien koon kasvaessa tarve hankkia tietoa hajautetuista, heterogeenisistä tietolähteistä, kuten tietämys- ja tietokannoista sekä erilaisista tiedostoista ja tietovarastoista, kasvaa. Ilman nopeaa yhteyttä useisiin hajautettuihin tietolähteisiin tiedonhankinta on sekä vaivalloista että kallista. Välittäjät ovat ohjelmamoduuleja, jotka mahdollistavat välityspalvelut tietolähteisiin ja linkittävät tietolähteitä sekä sovellusohjelmia yhteen. Critchlowin & al. (1998) mukaan välittäjät ovat tietovarastojen kriittisimpiä komponentteja, koska ne muuntavat dataa lähdetietomuodosta tietovarastojen hyväksymään muotoon. Välittäjien toinen funktio on ylimääräisen tiedon poistaminen välitettävästä tietojoukosta. Välittäjien suorittamat operaatiot vähentävät välitettävää tietomäärää ja täten säästävät käyttäjän aikaa suodattamalla oleellisen tiedon suuremmasta tietomäärästä. Kääreet ovat ohjelmamoduuleja, joiden tehtävänä on poimia ja välittää tietoa tietolähteistä. Kääreet voivat myös muuntaa tietolähteistä poimittua tietoa eri muotoon. Käyttäjän välittämät tiedonhaut välitetään usein kääreille sellaisessa muodossa, että kyselyyn vastaaminen ei onnistuisi ilman haetun vastauksen muokkaamista.

Deitelin & al. (2001) mukaan XML on teknologia, jonka avulla kuvataan rakenteista tietoa virtuaaliympäristössä. XML mahdollistaa tagien käytön, joiden avulla tiedonkuvaus on tarkka ja tiedon rakenteen kuvaaminen on mahdollista. XML mahdollistaa tiedon muuntamisen ja siirtämisen verkossa sekä tietojen varastoinnin tiettyjen tekniikoiden ja sovellusten avulla. Gardarinin & al. (1999) mukaan Internet/intranet -alustat tukevat olemassa olevien tiedostojen, tietokantojen, tietovarastojen ja perintösovellusten (legacy applications) saatavuutta yhdeltä palvelimelta. XML mahdollistaa tietolähteiden integroinnin tiettyjen tekniikoiden avulla. Tiedon koostaminen ja tietokantojen integrointi XML:n avulla on edullista moniin muihin tekniikoihin verrattuna.

Luvussa 2 tarkastellaan yhdistettyjä tietokantajärjestelmiä, yhdistämisen periaatetta ja tietokantajärjestelmän arkkitehtuuria. Luku 3 käsittelee tietovarastointia tiedon integrointikeinona. Luvussa 4 tarkastellaan välittäjäohjelmien käyttöä tiedon välityksessä eri tietolähteiden kesken ja kääreiden osallisuutta tiedon integrointiprosessiin. Luku 5

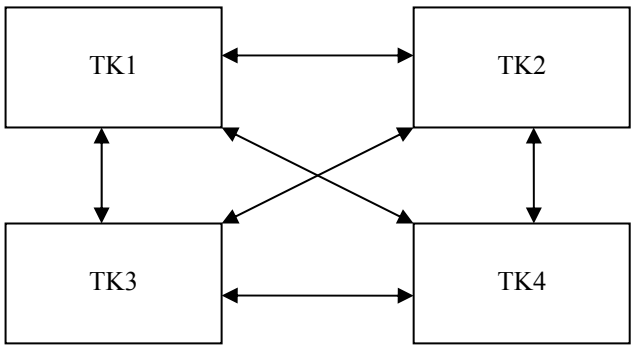
käsittelee XML:n käyttöä lähdetiedon yhdistämisessä. Luvussa 6 on yhteenveto käsitellyistä asioista.

2 YHDISTETTY TIETOKANTAJÄRJESTELMÄ

Yhdistetty tietokantajärjestelmä (federated database system, FDBS) tarkoittaa kokoelmaa yhdistettyjä tietokantoja ja niiden käytön mahdollistavia tietokannan hallintajärjestelmiä (database management system, DBMS). Tietokannan hallintajärjestelmän avulla yhdistettyyn tietokantajärjestelmään kuuluvat tietokannat voivat välittää informaatiota toisilleen. Yhdistettyyn tietokantajärjestelmään kuuluvat tietokannat ovat itsenäisiä, mutta yksi tietolähde voi kutsua toisia lähteitä hankkiakseen informaatiota (Sheth ja Larson, 1990).

2.1 Yhdistämisen periaate

Yhdistettäessä useita tietokantoja luodaan yhteys kaikkien niiden tietokantojen välille, joiden on kyettävä kommunikoimaan keskenään. Yhteys mahdollistaa tietokantajärjestelmän TK1 suorittaa kysely tietokantajärjestelmälle TK2 siten, että TK2 ymmärtää kyselyn. Tällaisen toteutuksen ongelmana on järjestelmään kuuluvien tietolähteiden määrä. Mikäli n kappaletta tietokantoja tarvitsee kommunikoida n-1:lle tietokannalle, on tätä varten kirjoitettava $n(n\text{-}1)$ yksikköä koodia tietokantojen välisten kyselyiden vuoksi. Seuraava kuva esittää neljän tietokannan muodostamaa yhdistettyä tietokantajärjestelmää (Garcia-Molina & al., 2000).



Kuva 1. Yhdistetty tietokantajärjestelmä (Garcia-Molina & al., 2000).

Kuvan 1 neljä tietokantaa muodostavat yhdistetyn tietokantajärjestelmän. Jokainen järjestelmään kuuluvasta neljästä tietokannasta tarvitsee kolme komponenttia kyselyiden suorittamiseen, eli yhden jokaista tietokantaa kohden.

Tarkastellaan esimerkkinä autojen myyntiketjua, joka haluaa inventoida varastossa olevat tuotteet. Jokaisen toimipisteen täytyy suorittaa tietokantakysely saadakseen selville millaisia autoja muiden varastot sisältävät. Oletetaan toimipiste TK1, jolla on kuvan 2 mukainen relaatio.

HalututAutot

Malli	väri	automaattiVaihteisto
200CD	valkoinen	TRUE
230E	valkoinen	TRUE
300E	sininen	TRUE

Kuva 2. Toimipisteen TK1 relaatio.

Relaation HalututAutot sarakkeina ovat asiakkaan pyytämä malli, väri ja automaattivaihteisto. Toimipisteellä TK2 on kaksi relaatiota sisältävä kaava kuvan 3 mukaisesti.

Autot

sarjaNumero	malli	väri
12093	200C	valkoinen
11051	230E	valkoinen
11033	260E	valkoinen

Valinnat

sarjaNumero	valinta
12093	manuaalivaihteisto
11051	automaattivaihteisto
11029	automaattivaihteisto

Kuva 3. Toimipisteen TK2 relaatiot.

Toimipiste TK1 kirjoittaa sovellusohjelman, joka suorittaa kyselyn toimipisteen TK2 varastossa olevista autoista, jotka ovat toimipisteen TK1 relaatiossa kuvattujen kaltaisia. Seuraava koodi kuvaa ohjelman osaa, joka suorittaa halutun kyselyn:

```
For (each tuple (:m, :v, :a) in HalututAutot) {
        If (:a = TRUE) {  /*halutaan automaattivaihteisto*/
                SELECT sarjaNumero
                FROM Autot, Valinnat
                WHERE Autot.sarjaNumero = Valinnat.sarjaNumero AND
                        Valinnat.valinta = 'automaattivaihteisto' AND
                        Autot.malli = :m AND
                        Autot.väri = :v
        }
        else {  /*automaattivaihteistoa ei haluta*/
                SELECT sarjaNumero
                FROM Autot
                WHERE Autot.malli = :m AND
                        Autot.väri = :v AND
                        NOT EXISTS (
                                SELECT *
                                FROM Valinnat
                                WHERE sarjaNumero = Autot.sarjaNumero
                                AND valinta = 'automaattivaihteisto'
                        );
        }
}
```

Kuva 4. Halutun kyselyn suorittama ohjelman osa.

Kuvan 4 kysely on toimipisteen TK2 tietokantajärjestelmän kaavan mukainen. Oletetaan, että toimipiste TK1 haluaisi suorittaa saman kyselyn toimipisteelle TK3, jolla olisi kuvan 5 mukainen relaatio.

Autot

sarjaNro	malli	Väri	automaattiVaihteisto
11090	220C	Sininen	TRUE
11060	230E	Sininen	TRUE
11025	300E	Sininen	TRUE

Kuva 5. Toimipisteen TK3 relaatio.

Tällöin kysely olisi erilainen, koska toimipisteiden relaatiot ovat erilaiset.

2.2 Yhdistetty tietokannan hallintajärjestelmä

Yhdistetty tietokannan hallintajärjestelmä (federated database management system, FDBMS) on ohjelmisto, joka mahdollistaa yhdistetyn tietokantajärjestelmän hallin-

nan. Ohjelmiston avulla tietokantajärjestelmään kuuluvat tietokannat voivat kommunikoida keskenään. Toisistaan erillään olevien tietokantojen ylläpito voi aiheuttaa hankaluuksia, koska erilaiset tietokannat saattavat olla identifioitu eri tavoin ja eri laitteistoilla voidaan käyttää erilaisia tietokantakieliä. Haettu tieto on myös siirrettävä pyynnön lähettäneelle laitteistolle, mikäli vastaus kyselyyn saadaan eri paikasta (Kamel ja Kamel, 1990).

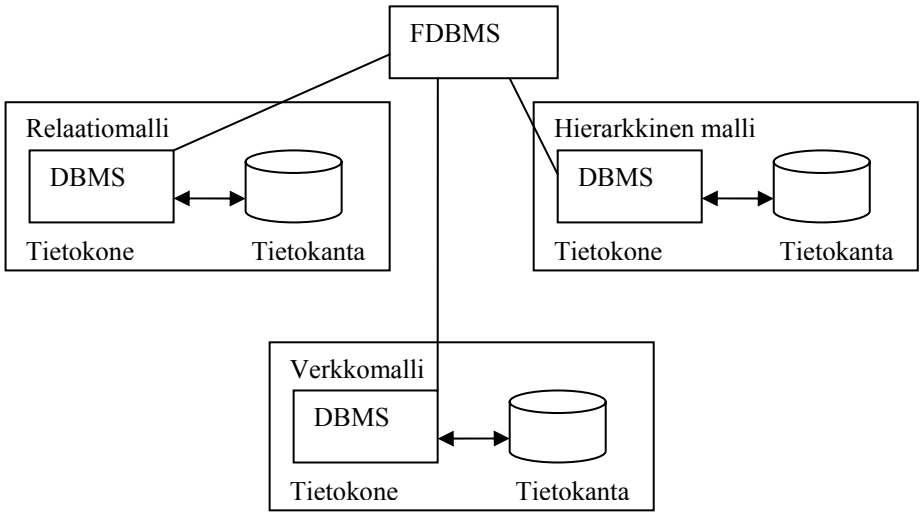
2.2.1 Ominaisuudet

Kamelin ja Kamelin (1990) mukaan yhdistetyltä tietokannan hallintajärjestelmältä vaaditaan ominaisuuksia, jotka määritellään seuraavasti:

1. Käyttäjän on päästävä useisiin heterogeenisiin tietokantoihin aivan yhtä helposti kuin käytettäessä yhtä tietokantaa. Järjestelmän on sallittava yhden tai useamman globaalin, integraation mahdollistavan kaavan määrittely. Kyselyn suorittaminen kaavan avulla mahdollistaa käyttäjän pääsyn heterogeenisen tietokannan sisältämään tietoon.
2. Käyttäjän on päästävä mihin tahansa tietokantaan käyttämällä tuttua tietomallia ja tietokantakieltä. Yhdistetyn tietokannan hallintajärjestelmän on mahdollistettava tiedon jakelu huolimatta tietokantojen heterogeenisuudesta, tietomalleista ja tietokantakielistä. Käyttäjältä ei saa vaatia uuteen datamalliin perehtymistä tai uuden tietokantakielen opiskelua.
3. Yhdistetty tietokannan hallintajärjestelmä ei saa edellyttää mitään muutoksia tehtäväksi olemassaoleviin tietokantajärjestelmiin tai -sovelluksiin. Järjestelmän kehittäminen lähtee ajatuksesta, että olemassaolevat tietokantajärjestelmät ja niihin liittyvät sovellukset säilytetään sellaisenaan.
4. Järjestelmän tulee mahdollistaa uuden tietokannan lisääminen järjestelmään. Yhdistetyn tietokannan hallintajärjestelmän on oltava laajennettavissa.
5. Käyttäjän on kyettävä sekä hakemaan tietoa että päivittämään tietokantaa. Tiedonkulun molempiin suuntiin on oltava mahdollista.
6. Yhdistetyn tietokannan hallintajärjestelmän suorituskyvyn tulee olla verrattavissa homogeeniseen hajautettuun järjestelmään. Järjestelmään kuuluvien tietokantojen

heterogeenisuuden ”piilottaminen” vaatii tietokannan hallintajärjestelmän tiedon prosessointitasojen lisäämistä, mikä tosin kuormittaa järjestelmää.

Seuraava kuva esittää yhdistettyä tietokannan hallintajärjestelmää, joka koostuu kolmesta tietokannasta ja tietokantojen hallintajärjestelmistä (Sheth & Larson, 1990).



Kuva 6. Yhdistetty tietokannan hallintajärjestelmä (Sheth ja Larson, 1990).

Kuvan 6 tietokantojen hallintajärjestelmät voivat kommunikoida toistensa kanssa yhdistetyn tietokannan hallintajärjestelmän avulla. Jokainen järjestelmään kuuluva tietokanta käyttää erilaista tietomallia. Tiedonvälitys on mahdollista tietomallien erilaisuudesta huolimatta.

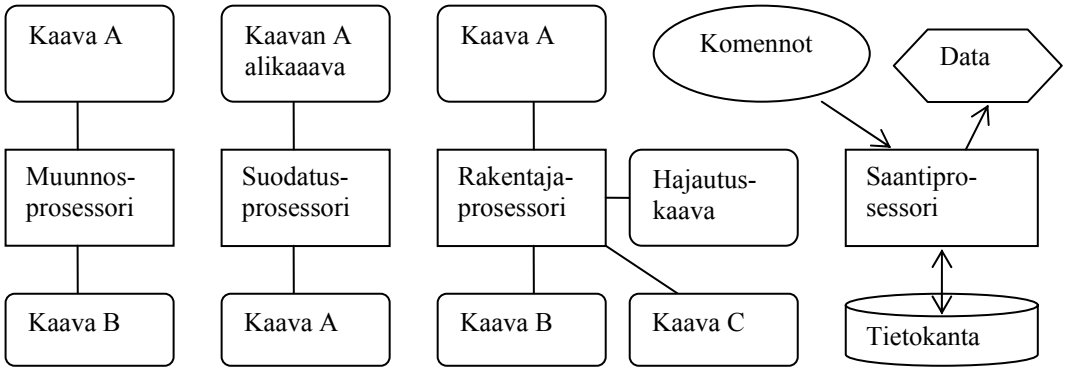
2.2.2 Arkkitehtuuri

Sheth ja Larson (1990) esittävät yhdistetyn tietokannan hallintajärjestelmän viitearkkitehtuurin, joka koostuu erilaisista järjestelmäkomponenteista. Komponentteja ovat data, tietokanta, komennot, prosessorit, kaavat ja kuvaukset.
Datan säilytyspaikkana toimii *tietokanta*, joka on rakenteeltaan tietomallin mukainen. *Komennot* ovat joko käyttäjän tai prosessorin antamia pyyntöjä suorittaa tiettyjä toimintoja. *Prosessorit* (processors) ovat sovelluksista riippumattomia tietokannan hallintajärjestelmiin kuuluvia ohjelmistomoduuleja, jotka käsittelevät komentoja ja dataa. *Kaavat* (schemas) ovat sovelluskeskeisiä komponentteja, jotka määrittelevät tietokan-

nan sisällön ja rakenteen. *Kuvaukset* (mappings) ovat funktioita, jotka asettavat kaavan alkiot vastaavuussuhteeseen toisen kaavan alkioiden kanssa.

Komponentteja voidaan yhdistellä eri tavoin, jolloin saadaan aikaan erilaisia tiedonhallinta-arkkitehtuureja. Kaksi peruskomponenttia, prosessorit ja kaavat, ovat tärkeitä erilaisten arkkitehtuurien määrittelyssä.

Prosessoreita on neljää eri tyyppiä, joista kukin suorittaa erilaisia toimintoja datalle ja komennoille: muunnosprosessorit, suodatusprosessorit, rakentajaprosessorit ja saantiprosessorit. Kuva 7 esittää nämä neljä prosessoria abstrakteina prosessoreina. Rakentajaprosessoria lukuunottamatta prosessorien toiminta käy ilmi myös kuvasta 8.



Kuva 7. Abstraktit prosessorit.

Muunnosprosessorit kääntävät lähdekieliset komennot kohdekielisiksi tai muuntavat dataa muodosta toiseen. Muunnosprosessorit tarjoavat tietoriippumattomuuden, jota kutsutaan tietomallin läpinäkyvyydeksi. Tämä tarkoittaa sitä, että tietyn prosessorin tietorakenteet ja komennot on kätketty muilta prosessoreilta. Kuvan 7 abstrakti muunnosprosessori voi kääntää kaavan A kuvausten mukaiset komennot kaavan B kuvausten mukaisiksi tai muuntaa kaavan B kuvauksen mukaisen datan kaavan A kuvauksen mukaiseksi.

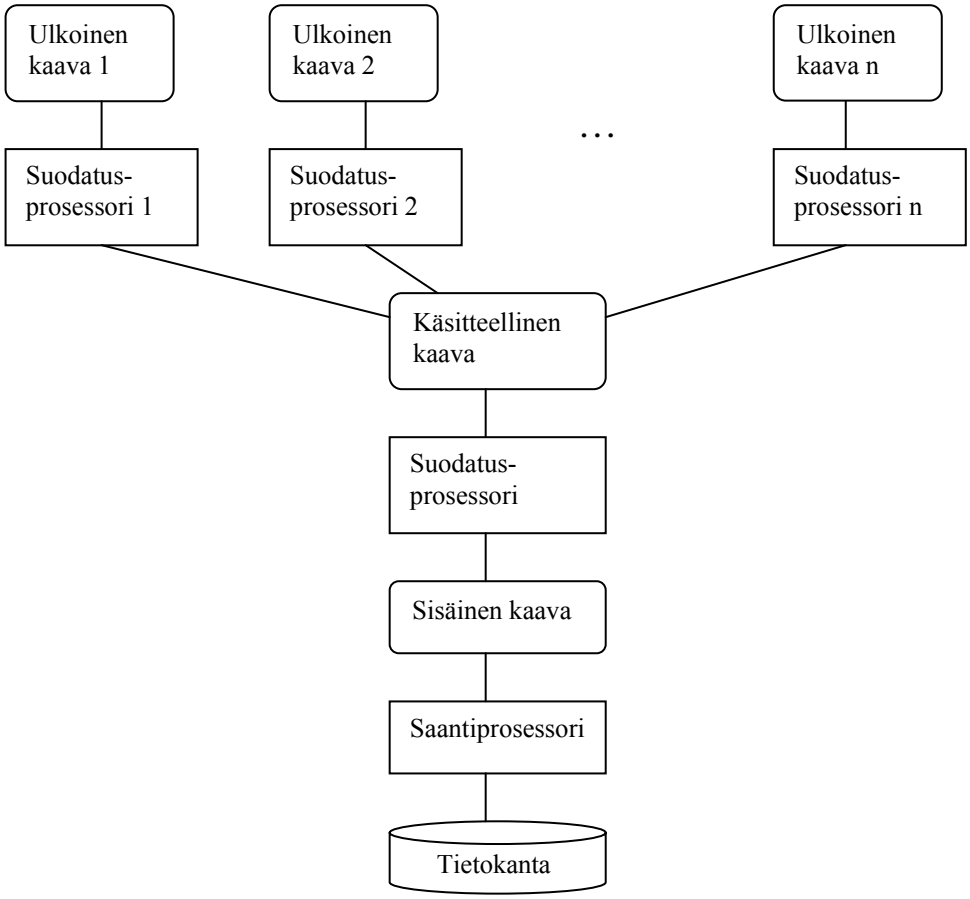
Suodatusprosessorit rajoittavat komentoja ja komentoihin liittyvää dataa, joita voidaan välittää toiselle prosessorille. Jokainen suodatusprosessori sisältää kuvaukset komentojen ja komentoihin liittyvän datan rajoitteista. Rajoitteet voidaan upottaa koo-

diin tai määritellä erillisessä tietorakenteessa. Kuvan 7 abstrakti suodatusprosessori voi suodattaa kaavan A tietorakenteiden mukaisen datan kaavan A alikaavan tietorakenteiden vaatimaan muotoon. Mahdollista on myös alikaavan komentojen suodatus kaavan A komentojen vaatimaan muotoon.

Rakentajaprosessorit osittavat ja/tai replikoivat yhden prosessorin suoritettavaksi tarkoitetut operaatiot sellaisiksi, että kaksi tai useampi muuta prosessoria hyväksyy ne. Rakentajaprosessorit myös liittävät useiden prosessoreiden tuottamaa dataa yhdeksi tietojoukoksi jonkin muun prosessorin käyttöön. Rakentajaprosessorit voivat tukea läpinäkyvyyttä paikan, hajautuksen ja replikoinnin suhteen. Kuvan 7 abstrakti rakentajaprosessori voi yhdistää kaavojen B ja C mukaisen datan kaavan A mukaiseksi dataksi. Mahdollista on myös kaavan A mukaisten komentojen jakaminen kaavojen B ja C tietorakenteiden mukaisiksi. Sekä datan yhdistämisessä että komentojen jakamisessa hyödynnetään kaavojen välisiä kuvauksia.

Saantiprosessorit hyväksyvät komentoja ja tuottavat dataa suorittamalla komentoja, joita välitetään tietokannalle. Saantiprosessori voi hyväksyä useiden muiden prosessoreiden komentoja ja limittää näiden komentojen suorituksen. Kuvan 7 saantiprosessori hyväksyy datan käsittelykomentoja ja käyttää saantimetodeja hakeakseen dataa tietokannasta.

Kaava koostuu kaava-alkioista (schema objects) ja niiden välisistä suhteista. ANSI/X3/SPARC -arkkitehtuuri on kolmen kaavatason kuvaus tietokannan hallintajärjestelmästä kuvan 8 mukaisesti. Tiedon kuvauksen kolme tasoa sisältävät käsitteellisen kaavan (conceptual schema), sisäisen kaavan (internal schema) ja ulkoisen kaavan (external schema) (Sheth ja Larson, 1990; Elmasri ja Navathe, 2000).



Kuva 8. ANSI/X3/SPARC –arkkitehtuuri.

Käsitteellinen kaava kuvaa käsitteelliset ja loogiset tietorakenteet sekä tietorakenteiden väliset suhteet. *Sisäinen kaava* kuvaa käsitteellisen kaavan sisältämien loogisten tietorakenteiden fyysiset piirteet. *Ulkoinen kaava* on käsitteellisen kaavan kaava-alkioista muodostettu alijoukko. Kolmen tason kaava-arkkitehtuuri sopii viitearkkitehtuuriksi keskittyihin tietokannan hallintajärjestelmiin (Sheth ja Larson, 1990).

Yhdistetyltä tietokannan hallintajärjestelmältä vaadittavien ominaisuuksien vuoksi yhdistetyn tietokantajärjestelmän kaava-arkkitehtuurin on sisällettävä viisi kaavatasoa kuvan 9 mukaisesti. Viiden tason arkkitehtuuri takaa riittävän tiedonkulun tietokantojen välillä, tietokantojen heterogeenisyyden ja tietokantajärjestelmien itsenäisyyden. Yhdistetyn tietokantajärjestelmän viiden tason kaava-arkkitehtuuri sisältää paikallisen kaavan (local schema), komponenttikaavan (component schema), vientikaavan (ex-

port schema), yhdistetyn kaavan (federated schema) ja ulkoisen kaavan (Sheth ja Larson, 1990).

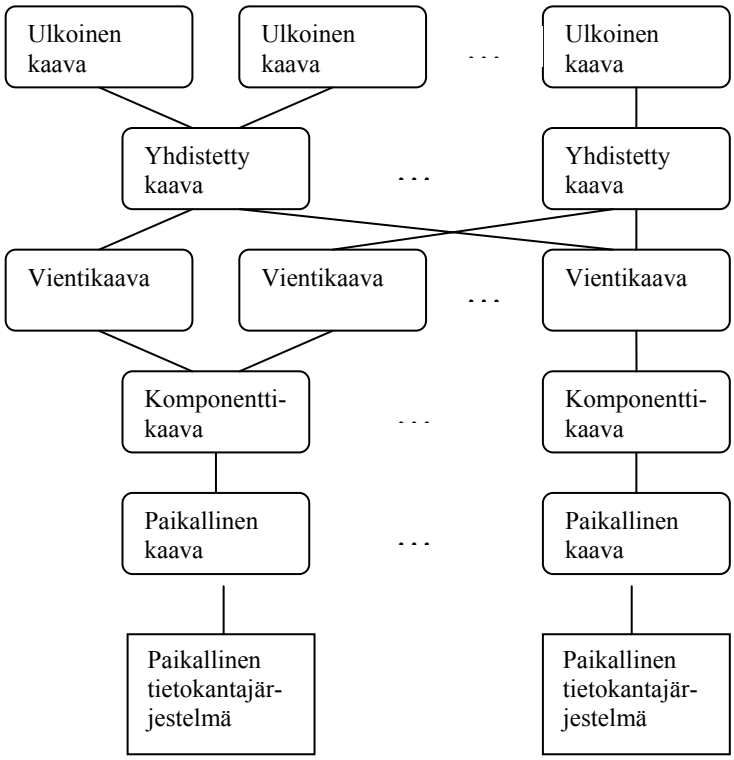
Paikallinen kaava on paikallisen tietokantajärjestelmän käsitteellinen kaava. Paikallinen kaava ilmaistaan paikallisen tietokannan hallintajärjestelmän tietomallin mukaisesti. Tämän vuoksi eri tietokantajärjestelmiin kuuluvat paikalliset kaavat voivat olla erilaisten tietomallien mukaisia, kuten kuvassa 6 esitettiin.

Komponenttikaavat johdetaan kääntämällä paikalliset kaavat yhdistetyn tietokantajärjestelmän yleisen tietomallin (common data model) mukaisiksi. Käännös tehdään, koska yleinen tietomalli kuvaa erilaisia paikallisia kaavoja yhden kuvaustavan mukaisesti ja paikallisesta kaavoista puuttuva semantiikka voidaan lisätä siihen liittyvään komponenttikaavaan. Paikallisen kaavan kääntäminen komponenttikaavaksi tuottaa komponenttikaava-alkioiden ja paikallisen kaavan kaava-alkioiden väliset kuvaukset. Muunnosprosessorit käyttävät näitä kuvauksia komponenttikaavaan kohdistuvien komentojen muuntamiseksi paikallisen kaavan mukaisiksi.

Kaikki paikallisen tietokantajärjestelmän tieto ei välttämättä ole yhdistetyn tietokantajärjestelmän ja käyttäjien saatavilla. *Vientikaava* edustaa komponenttikaavan alijoukkoa. Vientikaava voi sisältää käyttöoikeuksiin liittyvää tietoa, jonka avulla rajoitetaan tiedon saatavuutta joidenkin käyttäjien tai käyttäjäryhmien suhteen. Vientikaavan määrittelyn tarkoitus on helpottaa yhdistetyn tietokantajärjestelmän valvontaa ja hallintaa.

Yhdistetty kaava on useiden vientikaavojen yhdiste. Yhdistetty kaava sisältää myös informaatiota tiedonjaosta, jota syntyy kaavojen yhdistämisprosessissa. Jotkut järjestelmät käyttävät erillistä hajautuskaavaa (distribution schema) tiedonjakoon liittyvän informaation säilyttämiseen.

Ulkoinen kaava määrittelee kaavan käyttäjälle ja/tai sovellukselle. Ulkoinen kaava voi määritellä kaavan myös käyttäjä- ja/tai sovellusluokalle. Ulkoisen kaavan tarkoitus on ANSI/X3/SPARC –arkkitehtuurin tapaan toimia rajoittavana alijoukkona: eheysrajoitteiden lisääminen ja pääsyn valvonta.



Kuva 9. Yhdistetyn tietokantajärjestelmän viiden tason kaava-arkkitehtuuri (Sheth ja Larson, 1990; Elmasri ja Navathe, 2000).

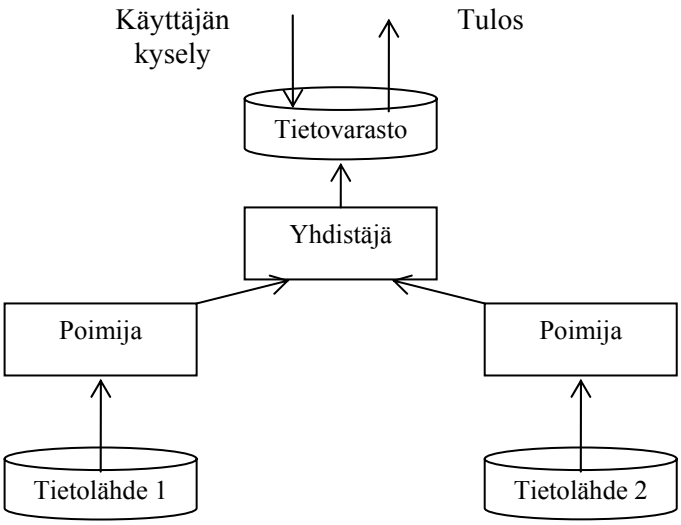
Kuvan 9 kaltaista yhdistettyä tietokantajärjestelmää määriteltäessä prosessorit sijoitetaan kaavojen väliin. Prosessorit modifioivat kaavojen välillä kulkevaa informaatiota. Yhdistetty tietokantajärjestelmä tarvitsee lisäksi tietokantajärjestelmän hoitajan, joka määrittelee ja organisoi kaavojen välisiä suhteita (Sheth ja Larson, 1990).

3 TIETOVARASTOINTI

Tietovarastointi on tiedon integrointiarkkitehtuuri, jossa useista hajautetuista, itsenäisistä ja mahdollisesti heterogeenisistä lähteistä poimittu tieto yhdistetään yhdeksi globaaliksi kaavaksi. Data varastoidaan tietovarastoon, joka näyttää käyttäjästä tavalliselta tietokannalta. Tietojen tallentamisen jälkeen käyttäjä voi kohdentaa kyselyitä tietovarastoon, kuten mihin tahansa tietokantaan (Garcia-Molina & al., 2000).

3.1 Tietovaraston periaate

Tietovarastoinnin tarkoitus on parantaa päätöksentekoa liiketoiminnassa. Furlowin (2001) mukaan päätöksentekoa varten tarvitaan historiatietoja, jonka vuoksi tietovarastosta ei yleensä poisteta mitään. Tietovaraston tietoja ei päivitetä, vaan tietovarastoon aina lisätään tarvittaessa uutta dataa. Seuraava kuva esittää tiedon poimimista tietolähteistä ja tiedon lataamista tietovarastoon.



Kuva 10. Tietovarsto, johon tallennetaan informaatiota erillisistä tietolähteistä (Garcia-Molina & al., 2000).

Kuvan 10 tietovarastoon tallennetaan integroitua informaatiota useasta lähteestä. Käyttäjien kyselyt kohdentuvat tietovarastoon. Tietovaraston tiedot kerätään tietolähteistä poimijan (extractor) avulla. *Poimija* kokoaa halutun tiedon tietolähteestä ja vä-

littää sen yhdistäjälle (combiner). *Yhdistäjä* integroi poimijoilta saadun tiedon ja lataa datan tietovarastoon.

Tarkastellaan esimerkkinä autoliikettä, jolla on kaksi toimipistettä. Oletetaan toimipiste 1, jolla on kuvan 11 mukainen relaatio ja toimipiste 2, jolla on kuvan 12 mukaiset relaatiot.

Autot

sarjaNro	malli	väri	automaattiVaihteisto	cdSoitin
10120	sedan	valkoinen	kyllä	kyllä
11121	sedan	musta	kyllä	kyllä
11122	coupe	punainen	ei	kyllä

Kuva 11. Toimipisteen 1 relaatio.

Ajoneuvot

sarjaNumero	malli	väri
10125	sedan	valkoinen
10139	sedan	punainen
10144	coupe	keltainen

Valinnat

sarjaNumero	valinta
10125	automaattiVaihteisto
10139	automaattiVaihteisto
10144	manuaaliVaihteisto

Kuva 12. Toimipisteen 2 relaatiot.

Oletetaan, että halutaan luoda tietovarasto, jolla on kuvan 13 mukaisen relaation mahdollistava kaava.

AutojenTiedot

sarjaNro	malli	väri	automaattiVaihteisto	toimipiste
10120	sedan	valkoinen	kyllä	toimipiste1
11121	sedan	musta	kyllä	toimipiste1
10144	coupe	keltainen	ei	toimipiste2
. . .				

Kuva 13. Tietovaraston relaatio.

Tässä tapauksessa globaali kaava sisältää kuvan 11 kaltaisen relaation. Tietovaraston luontia varten tarvitaan kuitenkin vain tieto siitä onko auto varustettu automaattivaihteistolla, sekä attribuutti, mikä kertoo toimittajan, jolla on kyseinen auto.

Koodi, joka erottaa halutun tiedon kahden toimipisteen tietokannasta voidaan kirjoittaa SQL-kyselyillä. Toimipisteen 1 poimijan SQL-kysely on kuvan 14 mukainen.

```
INSERT INTO AutojenTiedot (sarjaNro, malli, väri,
        automaattiVaihteisto, toimipiste)
    SELECT sarjaNro, malli, väri, automaattiVaihteisto, 'toimipiste1'
    FROM Autot;
```

Kuva 14. Toimipisteen 1 SQL-kysely.

Toimipisteen 2 poimija on monimutkaisempi, koska on päätettävä valitaanko auto automaattivaihteistolla vai ei. Tietotyyppinä käytetään merkkijonoa, jonka arvona on 'kyllä' tai 'ei'. Toimipisteen 2 poimija ilmaistaan SQL-kyselyllä, joka on kuvan 15 mukainen.

```
INSERT INTO AutojenTiedot (sarjaNro, malli, väri,
        automaattiVaihteisto, toimipiste)
    SELECT sarjaNumero, malli, väri, 'kyllä', 'toimipiste2'
    FROM Ajoneuvot, Valinnat
    WHERE Ajoneuvot.sarjaNumero = Valinnat.sarjaNumero
    AND valinta = 'automaattiVaihteisto';

INSERT INTO AutojenTiedot (sarjaNro, malli, väri,
        automaattiVaihteisto, toimipiste)
    SELECT sarjaNumero, malli, väri, 'ei', 'toimipiste2'
    FROM Ajoneuvot
    WHERE NOT EXISTS (
        SELECT *
        FROM Valinnat
        WHERE sarjaNumero = Ajoneuvot.sarjaNumero
        AND valinta = 'automaattiVaihteisto'
        )
```

Kuva 15. Toimipisteen 2 SQL-kysely.

Tietolähteistä erotettua tietoa ei tarvitse tässä tapauksessa yhdistää. Tietovarasto on tietolähteistä erotettujen relaatioiden yhdiste, joten tässä esimerkissä data ladataan

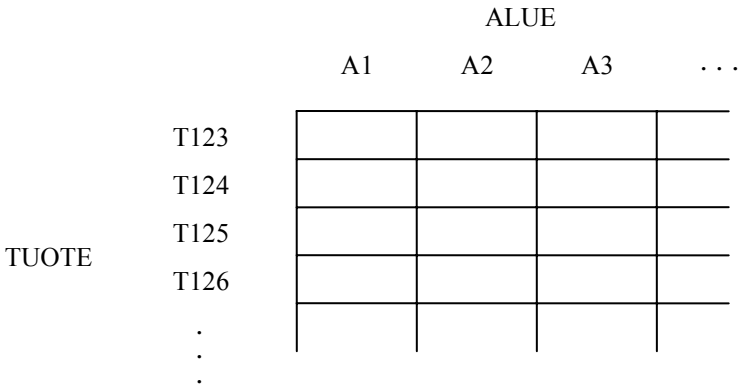
suoraan tietovarastoon. Jotkut tietovarastot suorittavat operaatioita tietolähteistä erotetuilla relaatioilla ennen tiedon lataamista tietovarastoon.

3.2 Arkkitehtuuri

Tietovaraston tietomallina käytetään usein moniulotteista mallia. Moniulotteinen tietomalli tukee päätöksenteon tuki- ja analysointiteknologioita. Moniulotteiset mallit käyttävät hyväkseen datan luontaisia suhteita tiedon latauksessa moniulotteisiin matriiseihin, joita kutsutaan tietokuutioiksi (data cubes). Moniulotteisen mallin mukaisessa tietovarastossa ulottuvuudet voisivat koostua yrityksen osavuosikatsauksista, tuotteista ja liiketoiminta-alueista, kuten kuvassa 17 (Elmasri ja Navathe, 2000).

3.2.1 Moniulotteisuus

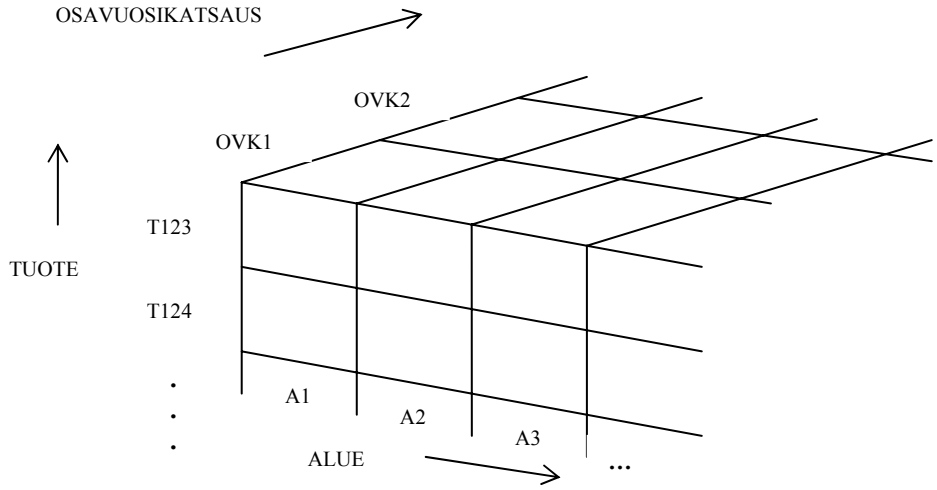
Standardi laskentataulukko on kaksiulotteinen matriisi. Eräs esimerkki laskentataulukon käytöstä olisi tiettyjen tuotteiden myynti alueittain tietyllä ajanjaksolla. Tuotteet voitaisiin esittää riveinä ja jokaisen alueen myynnit sarakkeina, kuten kuvassa 16. Aikaulottuvuuden lisääminen tuottaa kolmiulotteisen matriisin, joka voidaan esittää tietokuution avulla.



Kuva 16. Kaksiulotteinen matriisi (Elmasri ja Navathe, 2000)

Kolmiulotteinen matriisi, eli tietokuutio järjestää tuotteiden myynnit aikaperiodeihin myyntialueittain. Useamman kuin kolmen ulottuvuuden esittäminen graafisesti on hankalaa. Tietosisältöön voidaan kohdistaa kyselyitä millä ulottuvuuksien kombinaatioilla hyvänsä. Ulottuvuuden hierarkiasta toiseen siirtyminen onnistuu *kiertoa* (pivo-

ting, rotating) käyttämällä. Tällöin kolmiulotteisessa tietokuutiossa vaihdetaan kahta ulottuvuutta keskenään. Kaksiulotteisessa matriisissa vaihdettaisiin rivit ja sarakkeet keskenään. Seuraava kuva esittää kolmiulotteista tietokuutiota (Elmasri ja Navathe, 2000).

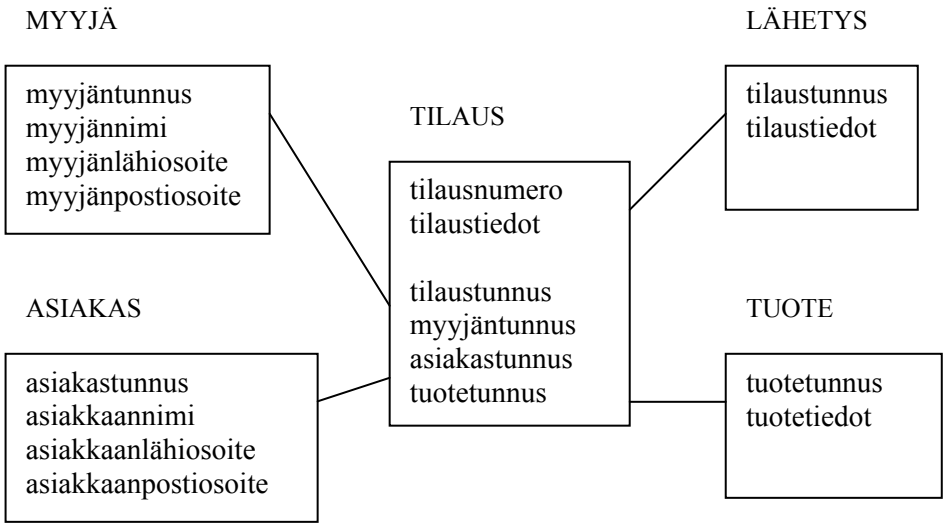


Kuva 17. Kolmiulotteinen tietokuutio (Elmasri ja Navathe, 2000).

Moniulotteisesta mallista saa muodostettua hierarkkisia näkemiä, joita kutsutaan karkeistavaksi operaatioksi (roll-up display) ja hienontavaksi operaatioksi (drill-down display). *Karkeistavassa operaatiossa* liikutaan hierarkiassa ylöspäin ja ryhmitetään tietoja suuremmiksi kokonaisuuksiksi, esimerkiksi summataan viikoittaisia myyntitietoja kuukausimyynneiksi. *Hienontavassa operaatiossa* suoritetaan päinvastainen toiminto, eli dataa jaetaan pienemmiksi yksiköiksi, esimerkiksi myyntitietoja saatetaan jakaa vaikkapa maakohtaisesta aluekohtaisiksi.

Tietovaraston moniulotteinen looginen malli koostuu kahdentyyppisistä tauluista: faktataulusta (fact table) ja ulottuvuustauluista (dimension tables). *Faktataulu* sisältää liiketoiminnan analysoitavat asiat, kuten myynnit. *Ulottuvuustaulut* sisältävät lisäpiirteitä, kuten tuotetunnukset tai tuotteiden hinnat. Ulottuvuustauluihin viitataan faktataulusta. Esimerkiksi faktataulun myynnit sisältämästä sarakkeesta myyntihinta viitattaisiin ulottuvuustaulun tuote vastaavaan riviin.

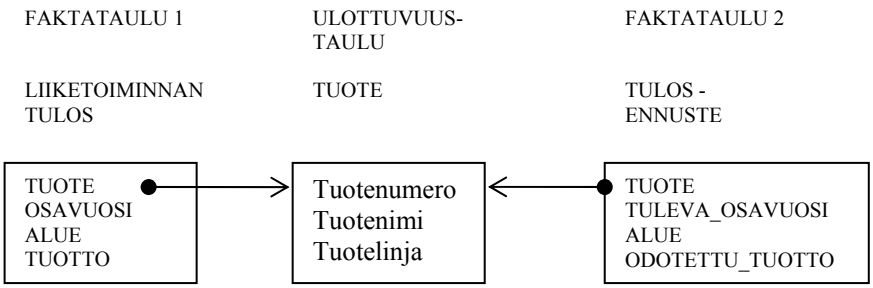
Kaksi yleistä moniulotteista kaavaa on tähtikaava (star schema) ja lumihiutalekaava (snowflake schema). *Tähtikaava* koostuu faktataulusta ja yhdestä taulusta jokaista ulottuvuutta kohden, kuten kuvassa 18. *Lumihiutalekaava* on tähtikaavan muunnos, jossa tähtikaavan ulottuvuustaulut ovat järjestetty hierarkkisesti. Järjestys määräytyy normalisoinnin mukaan. Useissa tietovarastosovelluksissa data normalisoidaan kolmanteen normaalimuotoon (Elmarsi ja Navathe, 2000).



Kuva 18. Tähtikaava fakta- ja ulottuvuustauluineen (Inmon, 1996).

Kuvan 18 tähtikaavan faktataululla tilaus on neljä ulottuvuustaulua, joihin faktataulusta viitataan.

Faktatauluryhmä (fact constellation) on joukko faktatauluja, jotka jakavat joitakin ulottuvuustauluja. Faktatauluryhmä rajoittaa tietovarastoon kohdennettavien kyselyiden määrää. Seuraava kuva esittää faktatauluryhmää (Elmasri ja Navathe, 2000).



Kuva 19. Faktatauluryhmä.

Kuvassa 19 faktatauluryhmä koostuu kahdesta faktataulusta: liiketoiminnan tulos ja tulosennuste, jotka molemmat viittaavat samaan ulottuvuustauluun.

3.2.2 Indeksointi

Tietovarasto mahdollistaa indeksointitekniikoiden hyödyntämisen suorituskyvyn parantamiseksi. Erityisesti tietovarastoihin liittyvä indeksoinnin toteutustapa on bittikarttaindeksi. *Bittikarttaindeksointi* (bitmap indexing) on tekniikka, jossa muodostetaan bittikartta sarakkeen jokaiselle erilliselle arvolle. Kukin bittikartta sisältää yhden bitin kutakin taulun riviä kohden. Seuraava kuva selventää bittikarttaindeksiä (Elmasri ja Navathe, 2000).

	Ostot	Bittikarttaindeksit	
Rivi	Aika_avain	1.1.2002	2.1.2002
1	1.1.2002	1	0
2	1.1.2002	1	0
3	2.1.2002	0	1
4	2.1.2002	0	1

Kuva 20. Bittikartan muodostaminen.

Bittikarttaindeksiä ilmentävässä kuvassa 20 tietokantataulun ostot sarakkeen aika-avain arvoille on muodostettu bittikartat. Jokaiselle sarakkeen eri attribuutin arvolle muodostetaan oma bittivektori. Bittikarttavektorien arvot ovat 1 ja 0. Arvo 1 vastaa arvoa kyllä ja arvo 0 arvoa ei.

Tähtikaavassa ulottuvuustaulujen data voidaan indeksoida siten, että faktataulusta viitataan ulottuvuustaulun riviin liitosindeksillä (join index). *Liitosindeksit* ovat perinteisiä indeksejä perus- ja viiteavainten välisten suhteiden ylläpitoon. Ne liittävät tähtikaavan ulottuvuustaulujen arvot faktataulujen vastaaviin riveihin (Elmasri ja Navathe, 2000).

Indeksoinnin ohella suorituskykyä voidaan parantaa *summatauluilla* (Hovi & al., 2003). Summataulu luodaan *materialisoitujen näkemien* avulla. Toteutunut näkemä

on kyselyn tulos. Tulos tallennetaan tietovarastoon, jolloin kyselyä ei tarvitse toistaa näkemän mukaisen vastauksen saamiseksi.

3.2.3 Paikallisvarastot

Tietovarasto sisältää yksityiskohtaista, geneeristä, toimialakohtaista dataa. Yrityksen toimiala- tai osastokohtainen tieto on hyödyllistä erottaa omaksi kokonaisuudekseen käsittelyn helpottamiseksi ja nopeuttamiseksi. *Datamartit* (data marts) eli *paikallisvarastot* ovat periaatteessa tietovarastojen toinen muoto ja niiden sisältämä tieto on tietovaraston sisältämän datan alijoukko (Sung ja Sang, 1998).

Paikallisvarastot ovat aihe- tai sovelluskeskeisiä suppeahkoja tietovarastoja. Paikallisvarastot sisältävät toimialakohtaista dataa, kuten myynti tai markkinointi. Tietovarastojen ja paikallisvarastojen ero on niiden sisältämässä informaatiossa. Lisäksi tietovarastot ovat usein huomattavasti laajempia kuin paikallisvarastot. Perinteisesti paikallisvarastot käyttävät tietolähteinään tietovarastoja, joten uuden tiedon lataus on nopeampaa kuin suuresta joukosta tietolähteitä tietovarastoon (Hobbs ja Hillson, 2000).

Paikallisvarastot voivat olla itsenäisiä. Riippumattomuus perustuu tietolähteeseen. Tieto ladataan itsenäisiin paikallisvarastoihin suoraan operatiivisista tietokannoista. Sitä vastoin paikallisvarastot, joiden tietolähteenä on tietovarasto, eivät ole riippumattomia.

3.3 Tietovaraston rakentaminen

Tietovaraston rakentamisessa on huomioitava tasapaino tapahtumankäsittelyn tehokkuuden ja käyttäjien suorittamien kyselyiden välillä. Tietovarasto optimoidaan tavallisesti tiedon analysoinnin tarpeisiin. Tämän vuoksi tietovaraston rakentamisessa on huomioitava seuraavat prosessit: tiedon varastointi tietomalli huomioon ottaen, tietorakenteiden luonti ja ylläpito, saantipolkujen luonti ja ylläpito, päivityksen tukeminen ja datan puhdistus (Elmasri ja Navathe, 2000).

3.3.1 Datalle asetetut vaatimukset

Tietovaraston rakentamisessa on otettava huomioon tietovaraston tuleva käyttö. Suunnitteluvaiheessa ei voi ennakoida kaikkia tulevia kyselyitä ja analyysejä, mutta suunnittelussa on huomioitava tiedon saatavuus kaikilla tarkoituksenmukaisilla attribuuttien arvoilla. Sopiva tietomalli on valittava tuleva käyttö huomioon ottaen. Tietovarastoon ladattavan tiedon suhteen on Elmasrin ja Navathen (2000) mukaan huomioitava seuraavia asioita:

1. Data on ladattava erillisistä, heterogeenisistä tietolähteistä, jotka saattavat sisältää hyvinkin erilaista ja erimuotoista tietoa.
2. Datan on oltava yhdenmuotoista tietovaraston tietojen kanssa. Nimet, tarkoitukset ja kohdealueet on sovitettava yhteen ja kaikki epäjohdonmukaisuudet on karsittava.
3. Data on puhdistettava kelpoisuuden varmistamiseksi. Datan puhdistaminen on monimutkainen prosessi. Puhdistamisen suunnittelu vaatii eniten työtä tietovaraston suunnittelu- ja rakennusvaiheessa. Datan puhdistamisen on tapahduttava ennen tiedon lataamista tietovarastoon. Puhdistuksen yhteydessä datan kelpoisuus ja laatu voidaan tarkastaa samalla kertaa. Myös tiedon saattaminen oikeaan muotoon voi tapahtua tässä yhteydessä. Puhdistus, kelpoisuuden ja laadun todentaminen sekä oikeaan muotoon muuntaminen kannattaa automatisoida niin pitkälle kuin mahdollista. Puhdistettu data voidaan palauttaa myös takaisin tietolähteeseen. Näin tietolähteiden datasta saadaan karsittua pois mahdollisia epäjohdonmukaisuuksia.
4. Data on sovitettava ja asennettava tietovaraston tietomallin mukaiseen muotoon. Dataa voidaan joutua muuntamaan relaatio- ja oliopohjaisista muodoista moniulotteiseen muotoon.
5. Data on ladattava tietovarastoon. Pelkkä tietovarastojen datan määrä tekee tiedon lataamisesta merkittävän toimenpiteen. Lataaminen vaatii tarkkailuvälineitä mahdollisesta virheestä johtuvasta lataustapahtuman keskeytyksestä toipumiseen. Tietovarastojen koon vuoksi ainoa käyttökelpoinen tapa tiedon latukseen on lisäävä päivitys. Päivitystavan suunnittelussa on huomioitava seuraavia seikkoja: kuinka usein tietoja on päivitettävä, voidaanko tietovarasto

sulkea ja kuinka pitkäksi aikaa, kuinka riippuvaista data on toisistaan, mikä on tietovaraston saatavuus, mitkä ovat tiedonjakelun vaatimukset ja mikä on latausaika.

Tietovaraston tiedot kuvataan *metatiedon* avulla (Elmasri ja Navathe, 2000). Metatieto voidaan jakaa kahteen osaan (Hovi & al., 2003): tekniseen ja käyttäjän eli liiketoiminnan metatietoon. Tietovarastoympäristöön on tarjolla useita *latausvälineitä* (ETL-tools). Niihin voidaan yleensä tallentaa tietovaraston lataukseen liittyvää, teknistä metatietoa. Tietovarastoihin tarjolla olevissa kysely- ja raportointivälineissä on monesti oma metatietojärjestelmänsä.

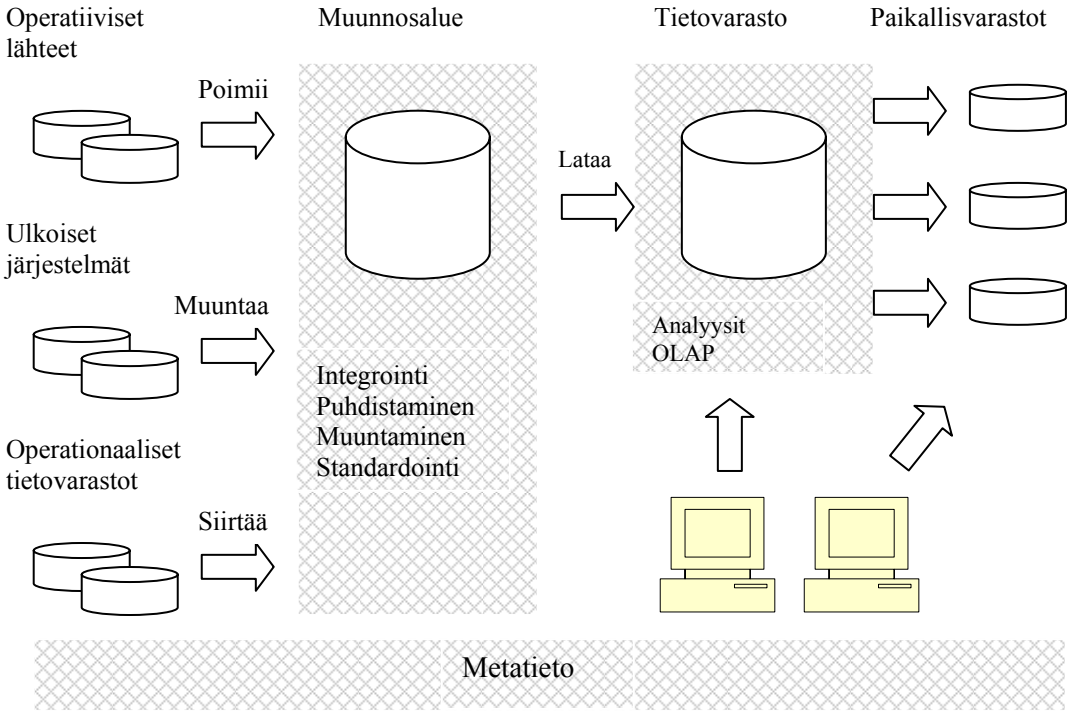
3.3.2 Datan puhdistaminen

Operatiivisten tietojärjestelmien data on erilaista kuin tietovarastojen data. Inmonin (1996) mukaan operatiivinen data on tietyssä tilassa olevaa, yritysten päivittäisiin operaatioihin käytettävää tietoa. Tietovarastojen data on informatiivista, historiatietoja sisältävää dataa.

Operatiivisista tietokannoista poimittu tieto voikin vaatia puhdistamista ja muuntamista toiseen muotoon. Tiedon puhdistaminen ja muuntaminen voi tapahtua erillisellä *muunnosalueella* (staging area). Muunnosalueena voi toimia esimerkiksi rakenteeton tiedosto (flat file). Puhdistettu ja muunnettu data voidaan ladata tietovarastoon tai suoraan paikallisvarastoihin.

Muunnosalueen käyttö edellyttää siis erillisen tilan varaamista. Tietolähteistä poimittu data siirretään väliaikaisesti muunnosalueelle, jossa se puhdistetaan, muunnetaan sopivaan muotoon ja valmistaudutaan lataamaan data tietovarastoon tai itsenäisiin paikallisvarastoihin. Tiedon puhdistamisella ja muuntamisella tarkoitetaan tiedon saattamista sellaiseen muotoon, että se voidaan ladata tietovarastoon. Poimitut tiedot saattavat olla eri tietotyyppiä, esimerkiksi päivämäärämuodot voivat olla erilaisia, tietoa saattaa puuttua tai tieto voi olla virheellistä. Lisäksi poimittu data saattaa sisältää samaa tietoa.

Seuraava kuva esittää tietovarastoarkkitehtuuria, joka koostuu tietovarastosta ja paikallisvarastoista tietolähteineen ja muunnosalueineen.



Kuva 21. Muunnosalue, tietovarasto ja datamartit (Hobbs ja Hillson, 2000).

Kuvassa 21 tieto poimitaan erilaisista tietolähteistä. Poimittu data siirretään erilliselle muunnosalueelle, jossa tieto puhdistetaan ja muunnetaan oikeaan muotoon ennen lataamista tietovarastoon. Muunnosalueella tietoa voidaan myös summata. Summaaminen on yhteenvetojen tekemistä olemassaolevasta datasta, esimerkiksi tietyn yksikön myynnit kuukausittain. Osa datasta siirretään erillisiin paikallisvarastoihin. Käyttäjä voi kohdentaa tiedonhaun joko tietovarastoon tai suoraan paikallisvarastoon. Metadata kuvaa prosessia datan poiminnasta käyttäjän tekemiin kyselyihin.

3.3.3 Datan lataaminen

Kun tiedot on poimittu tietolähteistä, puhdistettu ja muunnettu oikeaan muotoon, tiedot ladataan erilliseen tietokantaan eli tietovarastoon. Garcia-Molinan & al., (2000) mukaan tiedon lataamiseen tietovarastoon on ainakin kolme lähestymistapaa.

Yleisin lähestymistapa on määräajoin tehtävä tietovaraston uudelleen rakentaminen. Tällöin uusin tieto ladataan tietolähteistä tietovarastoon. Tiedonlataus voidaan suorit-

taa kerran vuorokaudessa, tosin tiedonlatauksen aikaväli voi olla pidempikin, esimerkiksi yksi viikko. Käyttäjän on tiedettävä kuitenkin tiedon ajantasaisuuden aste. Tiedonlataus suoritetaan yleensä öisin, jolloin tietovaraston käyttö on vähäistä. Tiedonlataamisen ajaksi järjestelmä on suljettava, tai järjestelmän käyttö on ainakin estetty. Tästä saattaa aiheutua ongelmia, mikäli tiedon lataaminen tietovarastoon kestää pidemmän aikaa.

Toinen tapa on tietovaraston päivittäminen määräajoin siten, että tietovarastoon ladataan tietolähteistä ainoastaan sellainen tieto, joka on muuttunut edelliseen päivitykseen nähden. Tässä lähestymistavassa käsiteltävän tiedon määrä voi olla suhteellisen pieni. Tiedon pienestä määrästä on etua siksi, että tiedonlataus voidaan suorittaa lyhyessä ajassa. Tällaista tiedonkeruuta nimitetään *lisääväksi päivitykseksi*. Haittapuolena on tietovaraston ja tietolähteiden vertailuun sopivien algoritmien suunnittelu, joka voi olla monimutkaisempaa kuin ensimmäistä lähestymistapaa käytettäessä.

Kolmas lähestymistapa on tietovaraston tietojen muuttaminen välittömästi, jos yksikin muutos tai joukko muutoksia tapahtuu yhdessä tai useammassa tietolähteessä. Tämä lähestymistapa vaatii liikaa kommunikaatiota ja tiedon prosessointia ollakseen käytännöllinen käytettäväksi kaikissa tietovarastoissa. Välitön tietojen päivittäminen sopii tietovarastoille, jotka ovat pieniä ja tietolähteiden, joiden tietoja käytetään tällaisten tietovarastojen päivittämiseen, on oltava hitaasti muuttuvia.

3.4 Toiminnallisuus

Tietovarasto rakennetaan käyttäjän tiedonsaantia varten. Tietovarastolla on oltava tehokas tuki käyttäjän suorittamien kyselyiden vuoksi. Tietovarastoon voi komponenttien avulla toteuttaa taulukkolaskennan toiminnallisuutta, rakenteisten ja käyttäjän suorittamien kyselyiden käsittelyä, analyysien tekoa ja materialisoitujen näkemien muodostusta.

Taulukkolaskennan toiminnallisuus sisältää tuen tietyille taulukkolaskennan sovelluksille, kuten MS Excel, sekä OLAP-sovellusohjelmille. OLAP-sovellukset sisältävät tiettyjä ennalta ohjelmoituja toiminnallisuutta lisääviä piirteitä. Yleisiä funktioita ovat

karkeistava ja hienontava analyysi, kierto, viipalointi (slice and dice), lajittelu, valinta ja johdetut attribuutit (derived attributes) (Elmasri ja Navathe, 2000).

Päätöksenteon tueksi tarvitaan erilaisia välineitä kyselyistä tietämyksen muodostukseen. Tietämyksen muodostamiseen käytetään tilastollisia menetelmiä. Taulukkolaskennan tai muun vastaavan sovellusohjelman avulla voidaan suorittaa tilastollista analyysiä. Tekoälytekniikoita, kuten geneettisiä algoritmejä tai neuroverkkoja, voidaan käyttää luokitteluun tai muuhun vastaavaan sellaisen tiedon tuottamiseen, jota ei kyselyiden avulla voida aikaansaada.

Tietovaraston voidaan ajatella olevan laajennus tietokannan näkemistä. Materialisoidut näkemät ovat eräs keino parantaa tiedon saantia. Ne tuottavat kuitenkin vain alijoukon tietovaraston datasta. Hovin & al. (2003) mukaan tietoa summataan usein esimerkiksi kuukausitasolle, mikä antaa todella hyvän vastausajan kuukausisummakyselyille. Summaaminen vie tosin levytilaa ja summataulujen rakentaminen vie aikaa.

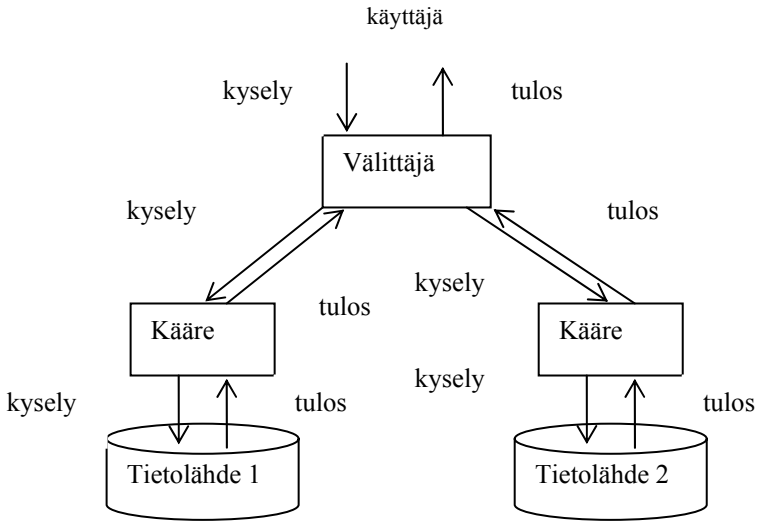
4 VÄLITTÄJÄT

Välittäjät ovat ohjelmistomoduuleja, jotka välittävät dataa sovellusten ja tietolähteiden välillä. Välittäjämoduulit suorittavat tiettyjä toimintoja, kuten muuntavat ja integroivat dataa tiedonvälityksen mahdollistamiseksi ja tiedon ylimäärän karsimiseksi. Välittäjät ovat tarpeellisia järjestelmissä, jotka käyttävät useiden ja erilaissa muodossa olevien tietolähteiden dataa (Wiederhold ja Genesereth, 1997).

4.1 Periaate

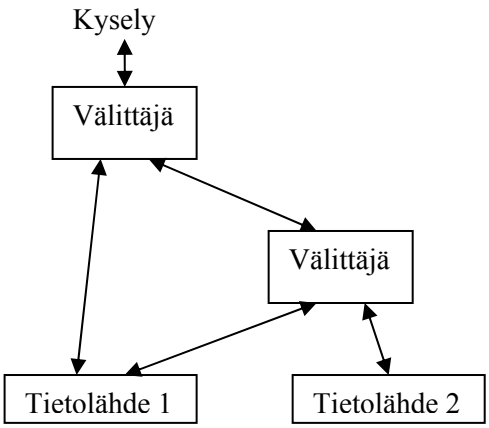
Välittäjät tukevat todellista näkemää tai näkemien joukkoa, joka yhdistää useita lähteitä samalla tavalla kuin materialisoidut relaatiot tietovarastossa integroivat lähteitä. Välittäjät eivät kuitenkaan varastoi dataa, joten välittäjien ja tietovarastojen toiminta on erilainen. Välittäjät voivat tosin säilyttää välituloksia (Garcia-Molina & al., 2000).

Kuva 22 esittää välittäjää, joka yhdistää dataa kahdesta lähteestä. Tietolähteitä voi olla enemmänkin kuin kaksi, kuten tyypillisesti tietovarastosovelluksissa on. Kuvassa käyttäjä suorittaa kyselyn välittäjälle. Välittäjät eivät varastoi dataa, joten välittäjän on hankittava tarvittava data tietolähteistä muodostaakseen vastauksen käyttäjän kyselyyn. Välittäjä välittää kyselyn kääreille, jotka tulkitsevat kyselyt lähteiden mukaisesti. Välittäjä yhdistää vastaukset.



Kuva 22. Välittäjän periaate (Garcia-Molina & al., 2000; Patel-Schneider ja Siméon, 2003).

Kuvan 22 välittäjä lähettää kyselyn molemmille kääreille, jotka puolestaan lähettävät kyselyn niitä vastaaville tietolähteille. Kääreet ovat ohjelmamoduuleja, jotka muuntavat kyselyt ja vastaukset oikean muotoisiksi. Jokaista tietolähdettä kohden on yksi kääre. Välittäjä voi lähettää kyselyn usealle kääreelle, ja voi olla suorittamatta kyselyä kaikilta kääreiltä. Saatuaan vastaukset lähettämiinsä kyselyihin, välittäjä yhdistää saamansa tulokset. Garcia-Molinan & al. (1997) mukaan välittäjä voi myös käyttää toista tai toisia välittäjiä datan välitykseen, kuten kuvasta 23 ilmenee.



Kuva 23. Välittäjien verkko ja tietolähteet.

Tarkastellaan periaatteen ymmärtämiseksi esimerkkinä samanlaista skenaariota kuin luvun 3 kohdassa 3.1, mutta välittäjiä käyttäen. Välittäjä integroi samat kaksi autoista koostuvaa tietolähdettä, kuten kuvissa 11 ja 12, yhdeksi näkemäksi. Näkemä on relaatio, joka sisältää seuraavan muotoisen relaatiokaavan:

AutotVälittäjä(sarjaNro, malli, väri, automaattiVaihteisto, toimipaikka)

Oletetetaan, että käyttäjä suorittaa välittäjälle kyselyn, joka koskee punaisia autoja. Kysely on kuvan 24 muotoinen.

```
SELECT sarjaNro, malli
FROM AutotVälittäjä
WHERE väri = ’punainen’;
```

Kuva 24. Käyttäjän kysely välittäjälle.

Välittäjä voi lähettää saman kyselyn molempien tietolähteiden kääreille. Toimipisteen 1 tietolähteen kääre kääntää kyselyn toimipisteen 1 kaavan mukaiseksi. Kuten kuvassa 11, toimipisteen 1 relaatiokaava on muotoa:

Autot(sarjaNro, malli, väri, automaattiVaihteisto, cdSoitin)

Toimipisteen 1 tietolähteen kääre kääntää SQL-kyselyn. Kääreen suorittama käännös on kuvan 25 muotoinen.

```
SELECT sarjaNro, malli
FROM Autot
WHERE väri = ’punainen’;
```

Kuva 25. Toimipisteen 1 tietolähteen kääreen suorittama käännös.

Vastaukseksi välittäjälle kääre palauttaa joukon sarjaNro-malli –pareja. Samaan aikaan toimipisteen 2 tietolähteen kääre kääntää saman kyselyn toimipisteen 2 kaavan mukaiseksi, kuten kuvassa 12. Toimipisteen 2 kaava on muotoa:

Ajoneuvot(sarjaNumero, malli, väri)
Valinnat(sarjaNumero, valinta)

Toimipisteen 2 tietolähteen kääre kääntää SQL-kyselyn. Kääreen suorittama käännös on kuvan 26 muotoinen.

```
SELECT sarjaNumero, malli
FROM Ajoneuvot
WHERE väri = ’punainen’;
```

Kuva 26. Toimipisteen 2 tietolähteen kääreen suorittama käännös.

Toimipisteen 2 tietolähteen kääre palauttaa välittäjälle joukon sarjaNumero-malli –pareja. Välittäjä voi muodostaa yhdisteen saamistaan parien joukoista ja palauttaa tuloksen käyttäjälle.

Välittäjällä on useita vaihtoehtoja tuloksen muodostamiseksi. Välittäjä voi lähettää kyselyn yhdelle tietolähteelle ja päättää palautetun tuloksen perusteella tarvitseeko sa-

ma kysely lähettää uudestaan jollekin toiselle tai useammalle tietolähteelle. Välittäjä voi varastoida välituloksen siksi aikaa, kunnes vastaus toiseen kyselyyn saapuu. Välittäjä voi myös palauttaa vain yhdeltä tietolähteeltä saamansa vastauksen (Garcia-Molina & al., 2000).

Välittäminen (mediation) on arkkitehtoninen käsite. Välitysmoduulin toteutustapa on vähemmän tärkeä kuin moduulin kyky toimia suuremmassa kontekstissa. Tehokkaan välityspalvelun on vähennettävä käyttäjäsovellukselle välitettävän datan ylimäärää siten, että tietosisältö säilyy muuttumattomana. Suurempi informaation määrä sisällytettynä pienempään määrään dataa lisää informaation tiheyttä (Wiederhold ja Genesereth, 1997).

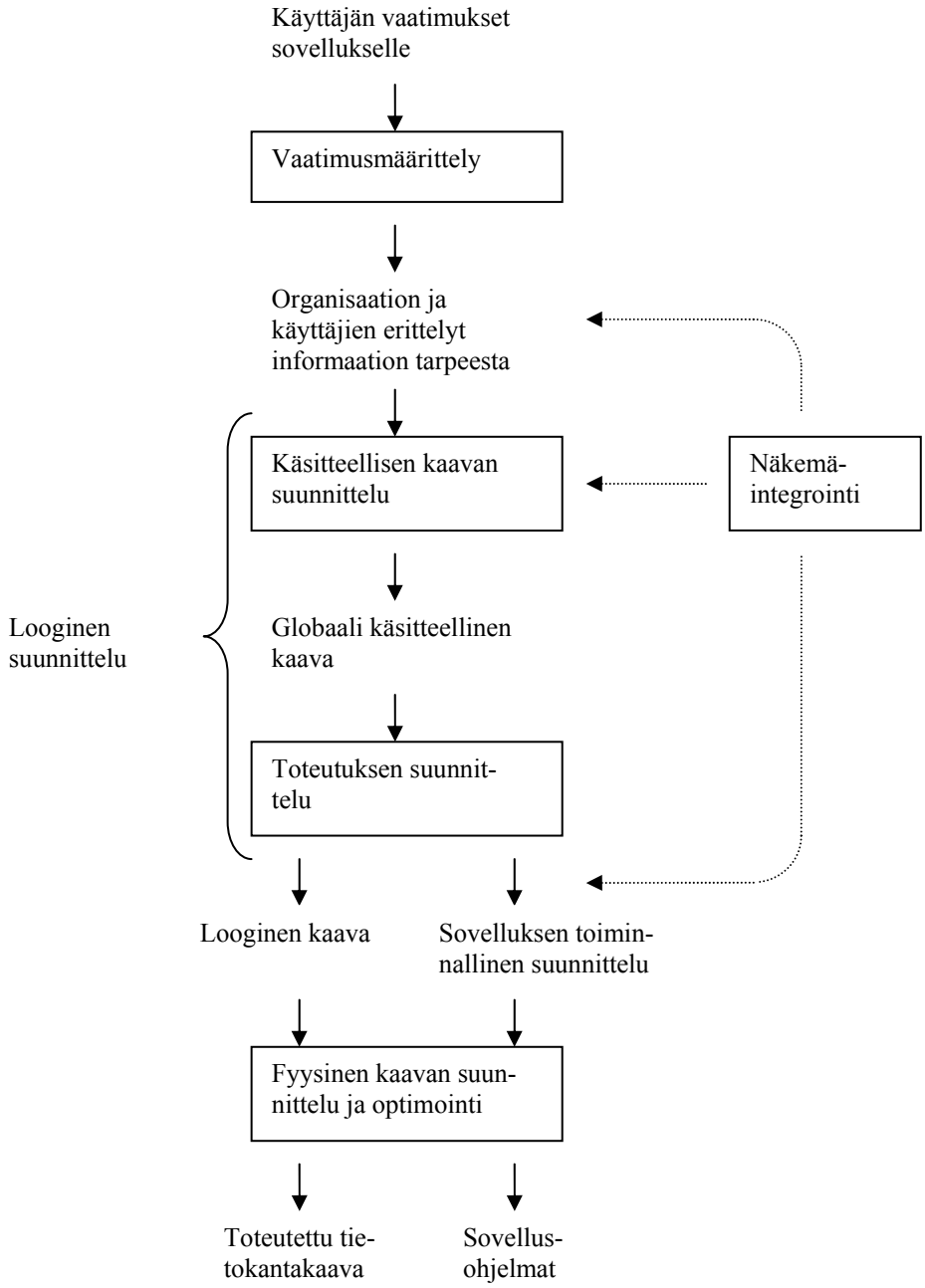
4.2 Tietolähteiden mallinnus

Tietolähteiden resurssimallin muodostamisessa yhdistyy Wiederholdin ja Geneserethin (1997) mukaan kolme käsitettä: kantakaavan integrointi, kaavaentiteettien yhdistäminen suhteiden avulla ja näkemien käyttö datan määrän vähentämiseksi. Jokainen dataresurssi voidaan kuvata omana mallinaan. Mallinnustekniikat eroavat toisistaan, joten tekniikoita voi joutua yhdistämään mallin aikaansaamiseksi.

Perinteisesti mallinnustekniikkana relaatiotietokantasuunnittelussa on ollut oliosuhdekaavio (Elmasri ja Navathe, 2000), mutta nykyisin myös UML–notaation mukaista luokkakaaviota käytetään käsitteellisessä suunnittelussa. Olioteknologiassa sovellusten ja tietokannan mallintamisen luonnollinen tapa on UML-notaatio oliomallin esittämiseksi (Connolly ja Begg, 2002).

Batini & al. (1986) soveltavat oliosuhdekaaviota tietokantakaavan integroimiseksi. Kuva 27 esittää suunnittelun vaiheistuksen, jossa näkemäintegrointi tuottaa kantakaavan. Näkemäintegrointi voidaan suorittaa prosessin eri vaiheissa, tavallisimmin kuitenkin käsitteellisessä suunnittelussa. El-Masri ja Wiederhold (1979) esittävät integrointiperiaatteen perustuen Boyce-Codd –normaalimuodossa oleviin relaatioihin. Tavoite on tuottaa integroitu tietokantakaava useampien käyttäjä- tai sovellusnäkemien

perusteella. Kantakaavan integrointi on Wiederholdin ja Geneserethin (1997) mukaan tarkoituksenmukaisempaa kuin tietokantojen integrointi, kuten luvussa 2 tarkasteltiin.



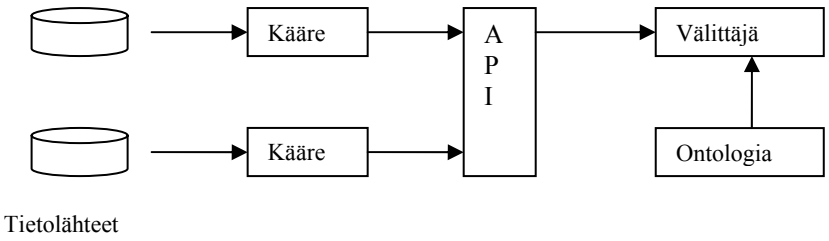
Kuva 27. Tietokannan suunnittelun vaiheet.

Merkkipohjainen data, kuten sanomat ja raportit, ovat myös tärkeä informaatioresurssi. Aiemmin merkkipohjaisen datan käsittely erotettiin tietokantatoiminnoista. Nykyisin muun muassa WWW-teknologia mahdollistaa merkkipohjaisen datan jaetun käsit-

telyn. Linkkien ja informaation integroinnin avulla rakenteista dataa voidaan liittää tekstiin. Linkit voivat osoittaa multimediaobjekteihin, kuten kuviin, ja nämä edelleen toisiin kohteisiin. Tällaiset dokumenttien sisäiset linkit eroavat tietokannassa käytettävistä yhteyksistä, eikä niitä voida mallintaa yleisen mallin avulla (Wiederhold ja Genesereth, 1997).

4.3 Ontologia

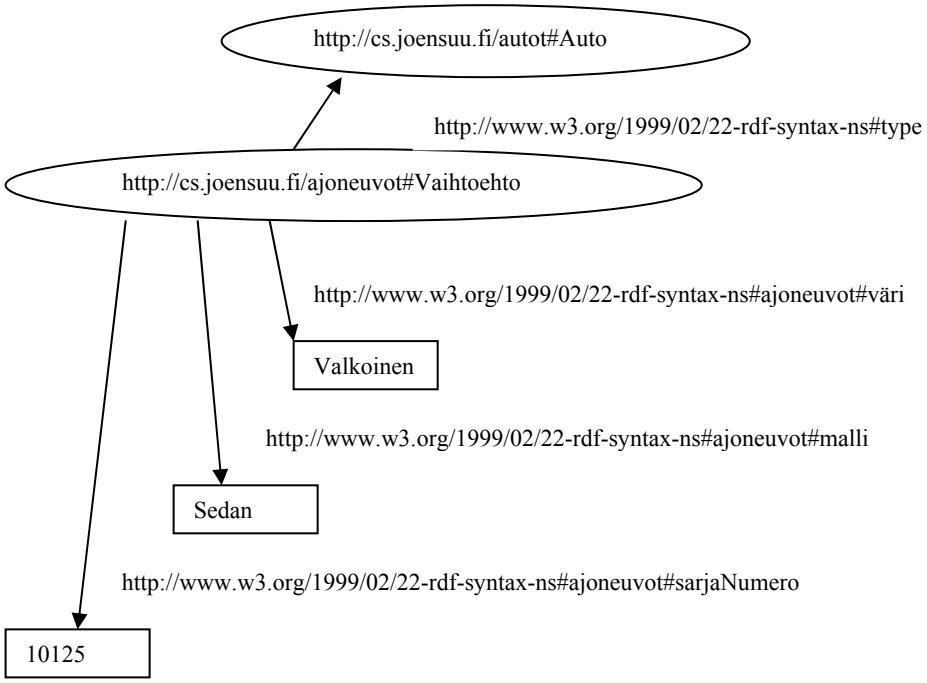
Critchlow & al. (1998) esittävät kuvan 28 mukaisen ratkaisun, jossa kääreiden ja välittäjien välillä on ohjelmointirajapinta, jota välittäjät käyttävät kutsuessaan kääreiden tarjoamia palveluja. Lisäksi kuvassa 28 esitetään ontologia, joka kuvaa tarvittavan metadatan.



Kuva 28. Rajapinta ja ontologia.

Ontologian käytön tavoitteena on Critchlowin & al. (1998) mukaan pystyä tuottamaan ja muuttamaan välittäjiä automaattisesti, kun tietolähteiden rakenteissa tapahtuu muutoksia.

Priebe ja Pernul (2003) käyttävät ontologiakuvauksessa RDF-graafeja ja RDF-kaavaa. RDF-graafi koostuu kolmikoista: resurssi (subjekti) linkitetään toiseen resurssiin (objekti) kaarella, jonka nimi muodostaa kolmannen resurssin (predikaatti). Resurssin sijaan kohteena voi olla literaali. Resurssin nimi on kvalifioitu URI (Uniform Resource Identifier). RDF-kaava koostuu luokista (class), jotka vastaavat geneeristä käsitettä tyyppi (type) tai kategoria (category) (Manola ja Miller, 2004b). Kuva 29 havainnollistaa RDF-graafia ja kuva 30 XML/RDF-kuvausta autoista. Kuvan 29 ellipsit ovat resursseja ja suorakaiteet literaaleja.



Kuva 29. RDF-graafi.

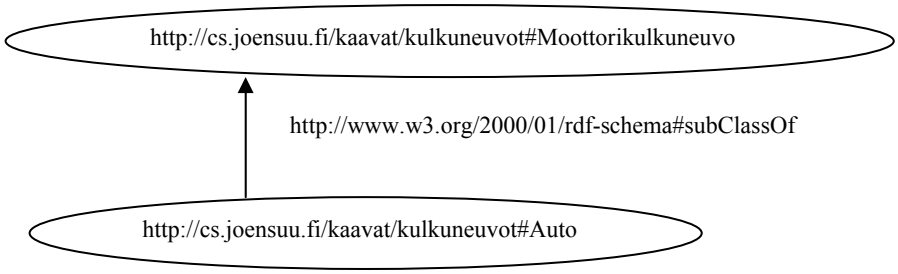
```
<?xml version=”1.0”?>
<rdf:RDF xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
                                        xmlns:autot=”http://cs.joensuu.fi/autot#”>

        <autot:Auto rdf:about=”http://cs.joensuu.fi/ajoneuvot#Vaihtoehto”>
                <ajoneuvot:sarjaNumero>10125</ajoneuvot:sarjaNumero>
                <ajoneuvot:malli>Sedan</ajoneuvot:malli>
                <ajoneuvot:väri>Valkoinen</ajoneuvot:väri>
        </autot:Auto>
</rdf:RDF>
```

Kuva 30. XML/RDF[1].

RDF-kaavan ja olio-ohjelmointikielen tyyppijärjestelmien välillä on yhteys. RDF-kaava mahdollistaa esimerkiksi resurssien määrittelyn luokkien ilmentyminä. Luokat voidaan myöskin järjestää hierarkkisesti. Kuvassa 31 on havainnollistettu Auto-luokkaa luokan Moottorikulkuneuvo aliluokkana. Kuva 32 esittää vastaavan kuvauksen RDF/XML-muodossa. W3C:n määrittämä OWL-kieli (Dean ja Schreiber, 2003) mahdollistaa RDF-kaavaan verrattuna kehittyneemmän tavan rajoitteiden määrittelemiseksi.

[1] XML-kieltä tarkastellaan tarkemmin luvussa 5.



Kuva 31. Luokkahierarkia.

```
<?xml version=”1.0”?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd ”http://www.w3.org/2001/XMLSchema#”>]>
<rdf:RDF
        xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#”
        xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#”
        xml:base=http://cs.joensuu.fi/kaavat/kulkuneuvot>

<rdf:Description rdf:ID=”Moottorikulkuneuvo”>
        <rdf:type rdf:resource=”http://www.w3.org/2000/01/rdf-schema#Class”/>
</rdf:Description>

<rdf:Description rdf:ID=”Auto”>
        <rdf:type rdf:resource=”http://www.w3.org/2000/01/rdf-schema#Class”/>
        <rdfs:subClassOf  rdf:resource=”#Moottorikulkuneuvo”/>
</rdf:Description>

</rdf:RDF>
```

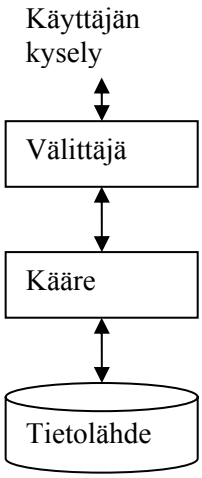
Kuva 32. RDF/XML.

RDF:n käyttö edellyttää ohjelmointirajapintoja. Verzulli (2001) ja Powers (2003) esittelevät Java-pohjaista ohjelmointirajapintaa. Powersin (2003) mukaan RDF/XML-rajapintoja on olemassa myös Perl-, PHP- ja Python-kielille.

4.4 Kääreet

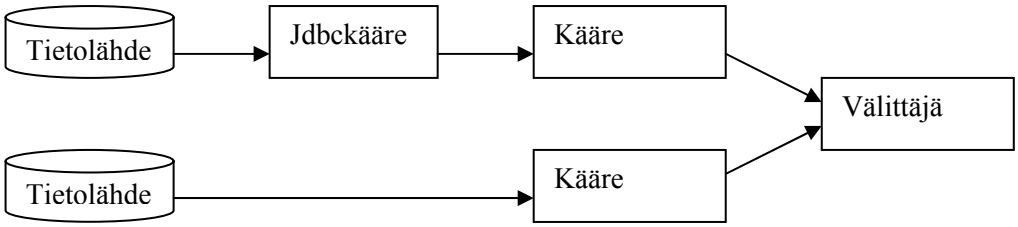
Kääreet mahdollistavat datan saannin heterogeenisistä tietolähteistä. *Kääre* on ohjelma, joka kääntää käyttäjän tietolähteille esittämät kyselyt tietolähteen ymmärtämään muotoon. Hearstin & al. (1998) mukaan jokainen kääre suunnitellaan tiettyä tietolähdettä varten. Tietolähteet voivat olla tekstitiedostoja, laskentataulukkoja tai data voi olla lähes missä muodossa tahansa. Garcia-Molinan & al. (1997) mukaan tietolähteek-

si käy esimerkiksi kysely tai komentojärjestelmä, kuten kirjastojärjestelmän hakukone. Kuva 33 havainnollistaa käärettä välittäjäarkkitehtuurissa.



Kuva 33. Tietolähde, kääre ja välittäjä välittäjäarkkitehtuurissa.

Välittäjän käyttämillä kääreillä voi olla omia kääreitä, esimerkiksi kuvan 34 mukaisesti JDBC-rajapinnan päälle rakennettu kääre (Laux, 2002; Travis, 2001).



Kuva 34. Tietolähteet, kääreet ja välittäjä.

Tietovarastosovelluksissa, kuten kuvassa 10, poimijat eli kääreet koostuvat yhdestä tai useammasta sisäänrakennetusta SQL-kyselystä, jotka suoritetaan datan hakemiseksi tietolähteistä. Lisäksi tarvitaan sopiva kommunikaatiomekanismi, jotta kääre voi välittää käyttäjän tekemiä SQL-kyselyitä tietolähteelle, vastaanottaa vastauksen tietolähteeltä ja välittää dataa tietovarastolle. Sisäänrakennetut kyselyt voivat olla SQL-kyselyitä, jos tietolähde on SQL-tietokanta. Kyselyt voivat olla myös suoritettavia operaatioita, jotka on toteutettu millä tahansa tietolähteen ymmärtämällä kannankäsittelykielellä (Garcia-Molina & al., 2000).

4.4.1 Mallinteet

Systemaattinen tapa rakentaa kääre, joka yhdistää välittäjän tietolähteeseen, on muodostaa erillinen mallinne. *Mallinne* (template) sisältää mahdolliset luokitellut kyselyt parametreineen, jotka edustavat vakioita. Välittäjä saa vakion mallinteelta ja kääre suorittaa kyselyn välittäjältä saamallaan vakioilla. Seuraava esimerkki havainnollistaa kuinka kääre kääntää mallinteen kyselyksi tietolähteelle (Garcia-Molina & al., 2000).

Esimerkkissä käytetään notaatiota *T* => *S*, jossa *T* = mallinne ja *S* = lähdekysely. Oletetaan, että halutaan rakentaa kääre tietolähteelle Toimipiste 1, jolla on kuvan 11 mukainen relaatiokaava, joka on muotoa:

Autot(sarjaNro, malli, väri, automaattivaihteisto, cdSoitin)

Toimipisteen 1 välittäjän relaatiokaava on muotoa:

AutotVälittäjä(sarjaNro, malli, väri, automaattivaihteisto, toimipiste)

Oletetaan, että välittäjän halutaan tiedustelevan tietyn värisiä autoja. Merkitään autojen väri parametrillä $v, jolloin kysely palauttaa halutun väriset autot. Mallinne on kuvan 35 muotoinen.

```
SELECT *
FROM AutotVälittäjä
WHERE väri = '$v';
        =>
SELECT sarjaNro, malli, väri, automaattivaihteisto, 'toimipiste1'
FROM Autot
WHERE väri = '$v';
```

Kuva 35. Mallinne, joka kuvaa tietyn väristen autojen kyselyä (Garcia-Molina & al., 2000).

Kuvan 35 mallinne palauttaa vastauksena halutun väriset autot. Autojen väri välitetään parametrina mallinteelle.

Kääreellä voisi olla toinen mallinne auton mallille, joka ilmaistaisiin parametrilla $m. Lisäksi kääreellä voisi olla vielä yksi mallinne, joka olisi automaattivaihteistoa varten. Tässä tapauksessa vaihtoehtoja on siis kahdeksan, mikäli kaikille kolmelle attribuutille (malli, väri, automaattivaihteisto) määritellään kysely. Yleisesti ottaen on olemassa 2^n mallinnetta, jos vaihtoehtoina on *n* kappaletta attribuutteja. Muita mallinteita voitaisiin tarvita esimerkiksi kyselyihin, joissa halutaan selvittää tietyn tyyppisten autojen kokonaismäärä tai selvitetään onko varastossa tietyn tyyppistä autoa. Kuvan 28 mukaisesti mallinteet voisivat muodostaa rajapinnan.

4.4.2 Suodattimet

Kääreiden rakentamisessa käytettävien mallinteiden määrä kasvaa suhteettoman suureksi, mikäli pyritään varautumaan kaikkiin mahdollisiin kyselyvaihtoehtoihin. Eräs tapa tukea suurempaa joukkoa SQL-kyselyitä on rakentaa kääresuodatin (wrapper filter) SQL-kyselyiden tulosten prosessoimiseksi. *Suodattimet* (filters) ovat ohjelmamoduuleja, jotka valikoivat ennalta määritellyin hakuehdoin dataa suuremmasta tietojoukosta. Suodattimet palauttavat alkuperäisen tietojoukon osajoukon (Garcia-Molina & al., 2000).

Oletetaan mallinne, kuten kuvassa 35, joka hakee tietokannasta tietyn värisiä autoja. Välittäjän halutaan hakevan sinisiä sedan –mallisia autoja, jolloin SQL-kysely on kuvan 36 muotoinen.

```
SELECT *
FROM AutotVälittäjä
WHERE väri = ’sininen’ and malli = ’sedan’;
```

Kuva 36. Välittäjän kysely sinisistä sedan –mallisista autoista.

Kuvan 36 SQL-kysely palauttaa kaikki tietokannassa olevat siniset sedan –malliset autot. Mahdollisuuksia vastata kyselyyn on:

1. Käyttää mallinnetta, kuten kuvassa 35, jolloin parametrissa $v välitetty auton väri määritellään muodossa $v = ’sininen’ kaikkien sinisten autojen löytämiseksi.

2. Varastoida tulokset väliaikaiseen relaatioon, jonka relaatiokaava on muotoa:

   VäliaikaisetAutot(sarjaNro,malli,väri,automaattiVaihteisto,toimipiste)

3. Valita relaatiosta VäliaikaisetAutot sedan –malliset autot ja palauttaa tulos SQL-kyselyssä, joka on kuvan 37 muotoinen.

```
SELECT *
FROM VäliaikaisetAutot
WHERE malli = 'sedan';
```

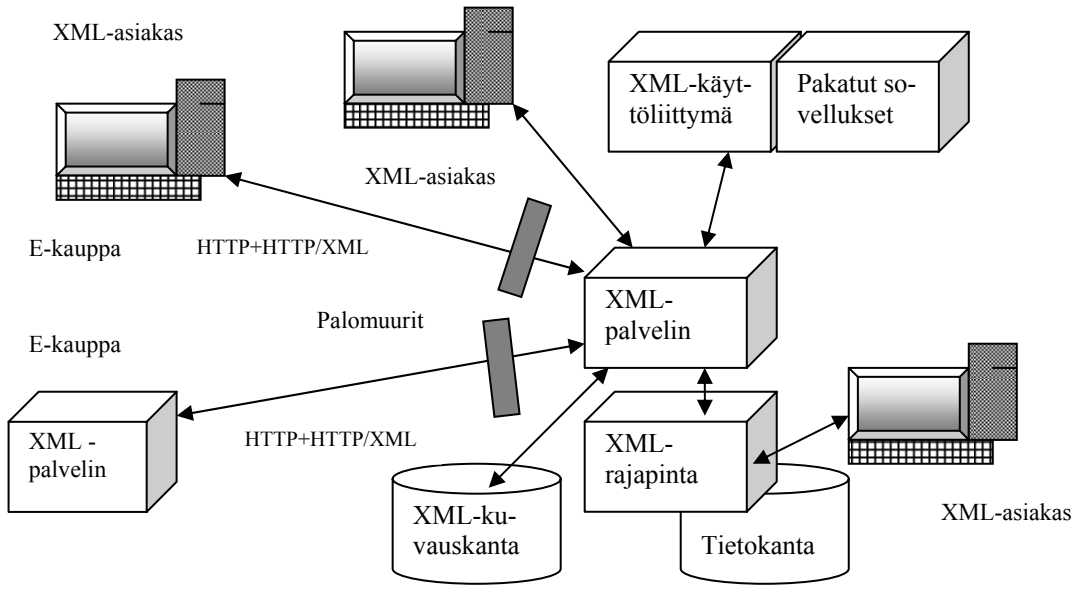
Kuva 37. SQL-kysely relaatiolle VäliaikaisetAutot (Garcia-Molina & al., 2000).

Kuvan 37 kysely palauttaa relaatiosta VäliaikaisetAutot sedan –malliset autot. Tuloksena on haluttujen autojen joukko. Käytännössä relaation VäliaikaisetAutot monikot voitaisiin tuottaa ja suodattaa yksi kerrallaan muodostamatta kääreelle materialisoitua relaatiota VäliaikaisetAutot.

Suodatinkomponentti on mahdollista sijoittaa muuallekin kuin kääreen yhteyteen. Kääre voi välittää raakadatan välittäjälle ja välittäjä suodattaa datan. Kuitenkin, jos suurin osa kyselyalustan palauttamasta datasta ei vastaa välittäjän kyselyä on parempi suorittaa suodatus kääreessä ja välttää näin välittämästä tarpeettomia monikoita.

5 XML

XML (Extensible Markup language) on tietojen kirjoittamisen avoin standardi, jonka avulla tietoja voidaan jakaa ohjelmistojen ja järjestelmien välillä. White & al. (2001) mukaan XML kehitettiin helpottamaan ohjelmistosovellusten käsittelyä Internetissä. XML:n käyttö ei kuitenkaan rajoittunut Internetiin, vaan XML:a käytetään laajalti erilaisissa ohjelmistosovelluksissa sekä lähdetiedon yhdistämisen apuna, kuten kuvasta 38 ilmenee.



Kuva 38. XML-teknologia yrityssovelluksissa (Lear, 1999).

Kuva 38 havainnollistaa, kuinka XML tarjoaa standardoidun tietomuodon datan siirtoon sekä yrityksissä että yritysten ja sovellusten välillä. Lisäksi XML-teknologia mahdollistaa eri muodossa olevien tietokantojen käytön erilaisilla laite- ja sovellusalustoilla.

5.1 XML-dokumentti

XML-kieli on sukua SGML-kielelle, joka on metakieli, eli kieli, jolla kuvataan merkkauskieliä (markup language). *Merkkaus* on koodijärjestelmä, jonka avulla määritellään dokumentin tekstinmuotoilu. XML tukee joukkoa piirteitä, kuten käyttäjän määrittelemät tagit, jotka sallivat datan ja datan kuvauksen esittämisen samassa dokumen-

tissa. Bertinon ja Catanian (2001) mukaan dokumentin tyyppimäärittely (document type definition) kuvaa tagit, joita dokumenteissa voidaan käyttää, sekä säännöt, jotka yhdistävät tagit dokumentin sisältöön. Dokumentin tyyppimäärittely, eli DTD, kuvaa myös kaikki ulkoiset elementit, joihin dokumentin sisällä on viitattu, sekä käytettävissä olevat notaatiot.

White'n & al. (2001) mukaan *tageiksi* kutsutaan kaikkia "pienempi kuin" –merkin (<) ja "suurempi kuin" –merkin (>) väliin tulevia tunnisteita, kuten <NAME>. Lisäksi merkkauksen voi osoittaa merkeillä "et" (&), heittomerkillä (') ja lainausmerkillä ("). Mikäli jotakin edellä luetelluista merkeistä halutaan käyttää normaaleina merkkeinä, ne tulee kiertää kirjoittamalla niitä vastaava merkkijono, eli entiteetti. XML-kielessä kaikki, mikä ei ole merkkausta on sisältöä, eli merkkidataa. Seuraava taulukko sisältää XML-kielen ennalta määritellyt yleisentiteetit.

Taulukko 1. Ennalta määritellyt yleisentiteetit (White & al., 2001).

Merkki	Korvaus
&	& tai &
'	' tai '
>	> tai >
<	< tai &>
"	" tai "

Taulukossa 1 nimetyn entiteetin voi kirjoittaa esittämään merkkiä, kuten > "suurempi kuin" -merkkiä. Lisäksi on mahdollista käyttää merkkiviittausta, kuten >.

Entiteetit ovat normaalisti ulkoisia objekteja, kuten kuvatiedostoja. Jos halutaan tehdä viittaus ulkoiseen objektiin, dokumentissa täytyy olla dokumenttityyppimäärittely. On olemassa myös toisenlaisia entiteettejä, joita voi käyttää. Tällaisia entiteettejä kutsutaan sisäisiksi entiteeteiksi. Sisäinen entiteetti on määriteltävä ennen käyttöä. Sisäisen entiteetin määrittely on muotoa: <!ENTITY nimi "korvaava teksti">. Aina kun &nimi esiintyy tekstissä, se korvataan heittomerkkien sisällä määritellyllä korvaavalla tekstillä, kuten kuvassa 39 riveillä 2-3 ja 6.

```
<?xml version=”1.0”?>
<!ENTITY rights ”All rights reserved including this product. No part of this book may be
reproduced or transmitted in any form without the prior permission of the publisher.”>
<asiakas>
        <tilausnumero=”2342312”>
        <teksti>© 1995, &rights;</teksti>
        <nimi>Peter Ylämö</nimi>
        <osoite>
                <katuosoite>Teljotie 43</katuosoite>
                <postinumero>80160</postinumero>
                <kaupunki>Joensuu</kaupunki>
        </osoite>
        <puhelin_numero>04001232020</puhelin_numero>
        <lähetyspäivä>02-02-2002</lähetyspäivä>
</asiakas>
```

Kuva 39. XML-dokumentti (Roy ja Ramanujan, 2001).

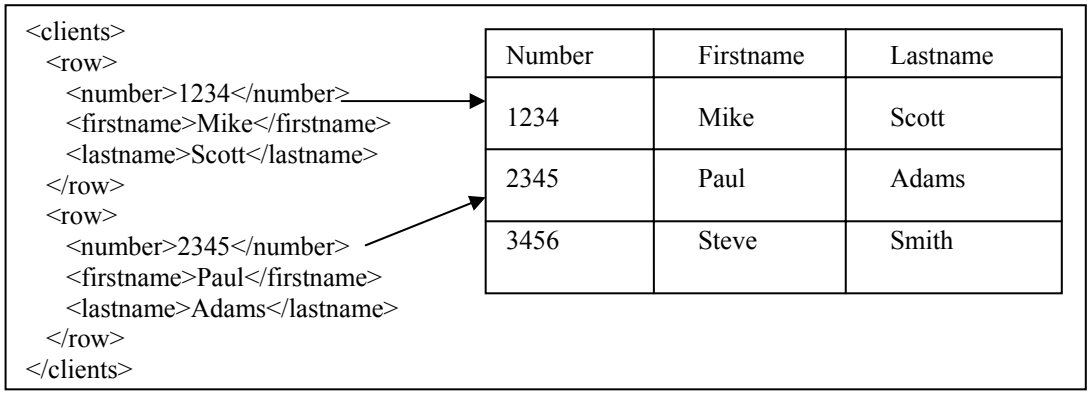
Kuvassa 39 rivillä 1 XML-määritelmä identifioi sitä seuraavan tekstin XML-koodiksi. Seuraavalla rivillä on määritelty sisäinen entiteetti *rights*, jonka avulla koodiin sijoitetaan *&rights* -merkkijonon tilalle entiteetissä määritelty korvaava teksti. Jokaisessa XML-dokumentissa on yksi juurielementti. Kaikkien muiden elementtien on oltava tämän juurielementin sisällä, kuten kuvassa 39 <asiakas>.

XML-kielen piirteet asettavat tietokannanhallintajärjestelmille tiettyjä vaatimuksia XML-dokumenttien sisältämän datan luettavuuden, varastoinnin, tiedonhaun ja datan yhdistämisen suhteen. Bertinon ja Catanian (2001) mukaan XML-dokumenttien hallittavuuden helpottamiseksi XML-dokumentit on tarkoituksenmukaista luokitella datakeskeisiin (data-centric) ja dokumenttikeskeisiin (document-centric) dokumentteihin.

5.1.1 Datakeskeinen XML-dokumentti

Tyypilliset datakeskeiset XML-dokumentit ovat säännöllisiä rakenteeltaan ja homogeenisiä sisällöltään, kuten kuvassa 39. Datakeskeiset XML-dokumentit liittyvät usein kaupallisiin sovelluksiin, joissa käsiteltävä tietovirta koostuu yksityiskohtaisista tuotetiedoista, laskutustiedoista tai esimerkiksi matkustusaikataulun sisältämästä tiedoista. Tällaisten dokumenttien sisältämä data on pääsääntöisesti tarkoitettu elektroniseen tiedonkäsittelyyn (Renner, 2001).

Tietokannan hallintajärjestelmien on tuettava datan erottamista (data extraction) ja datan muodostusta (data formatting) datakeskeisten XML-dokumenttien käsittelyn mahdollistamiseksi. Datan erottamista varten XML-dokumenttien ja tietokannan hallintajärjestelmän tietomallin välille on määriteltävä kaava. Rennerin (2001) mukaan yksinkertaisinta kaavan määrittelyä kutsutaan dokumentin tyyppimäärittelyksi (DTD). Dokumentin tyyppimäärittely mahdollistaa dokumentin rakenteen määrittelyn. XML-dokumentista erotettu data varastoidaan usein tauluihin, kuten kuvassa 40. Datan varastointi tauluihin ei säilytä XML-dokumentin alkuperäistä rakennetta. Taulumuotoisesta materialisoidusta näkemästä puuttuvat esimerkiksi tagit ja mahdolliset koodiin sisältyneet kommentit (Bertino ja Catania, 2001).



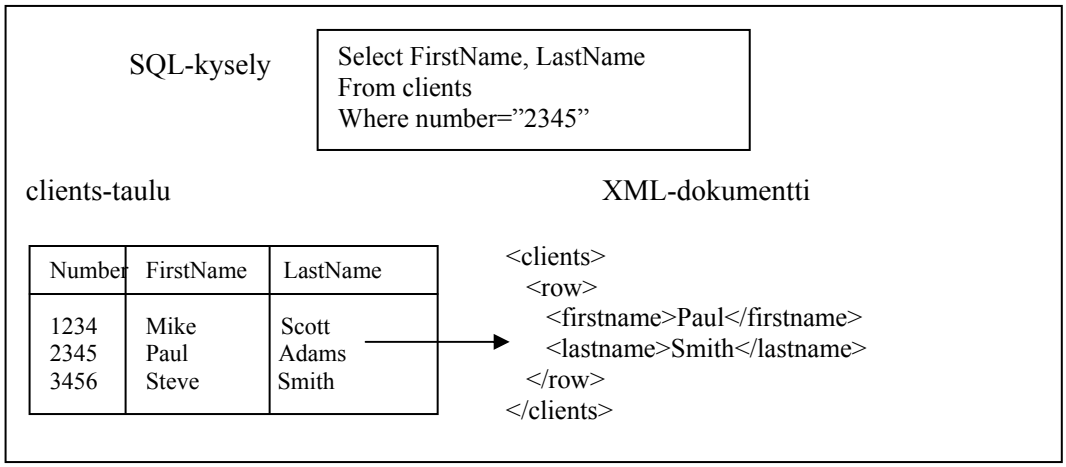
Kuva 40. XML-dokumentista erotettu data.

Kuvassa 40 XML-dokumentista taulukkomuotoon erotettua dataa voidaan käsitellä tietokannan hallintajärjestelmän kyselykielellä, kuten SQL. Relaatiopohjaisissa tietokannan hallintajärjestelmissä XML-dokumentti tulkitaan kuvan 41 mukaisen kaavan mukaan.

```
<database>
    <table>
      <row>
        <column1>…</column2>
        <column1>…</column2>
          …
    </row>
          …
  </table>
          …
</database>
```

Kuva 41. XML-dokumentin tulkinta relaatiokannassa (Bertino ja Catania, 2001).

Datan muodostusta tukevan tietokannan hallintajärjestelmän avulla XML-datasta muodostetuista tauluista voidaan erottaa SQL-kyselyllä halutut rivit ja muodostaa kyselyn palauttamasta vastauksesta XML-dokumentti, kuten kuvassa 42.



Kuva 42. Datan muodostus (Bertino ja Catania, 2001).

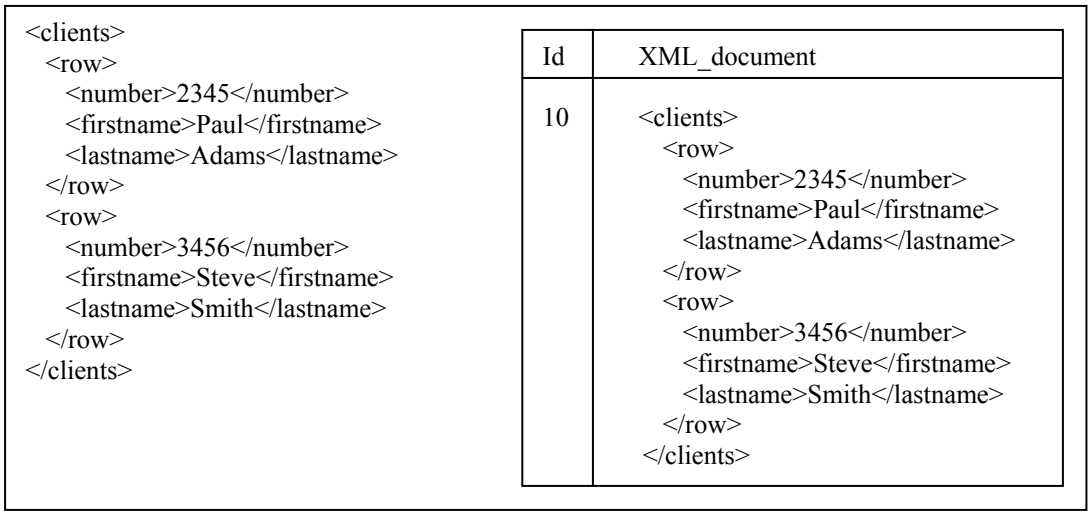
5.1.2 Dokumenttikeskeinen XML-dokumentti

Tyypillinen dokumenttikeskeinen XML-dokumentti sisältää suurimmaksi osaksi rakenteetonta dataa. Tällaisia dokumentteja ovat esimerkiksi lehtiartikkelit, sähköpostiviestit, kirjat ja lehdistötiedotteet sekä XHTML-kielellä laaditut Web-sivut. XHTML-kieli on XML- ja HTML-kielten yhdistelmä, joka yhdistää sekä dokumentin rakenteen että ulkoasun määrittelyn. Dokumenttikeskeiset XML-dokumentit voivat sisältää lisäksi esimerkiksi kuvia, mutta dokumentissa voi olla jonkin verran myös rakenteista dataa (Renner, 2001).

Bertinon ja Catanian (2001) mukaan dokumenttikeskeisten dokumenttien käsittely vaatii tietokannanhallintajärjestelmältä sopivia tietotyyppejä XML-dokumenttien esittämiseen tietokantojen sisällä sekä kyvykkyyttä kyselyiden muodostamiseen. XML-datan esittämiseen on kehitetty kaksi esitystapaa: rakenteeton esitystapa (unstructured representation) ja hybridi esitystapa (hybrid representation).

Rakenteettomalle esitystavalle on luonteenomaista käsitellä XML-dataa yhtenä objektina, esimerkiksi yhtenä tietokenttänä (single data field), kuten kuvassa 43. Doku-

mentti voidaan varastoida tietokantaan, tai varastointi voidaan tehdä tietokannan ulkopuolelle. Jos dokumentti varastoidaan muualle kuin tietokantaan, on dokumenttiin luotava linkki.



Kuva 43. XML-varasto käyttäen rakenteetonta esitystapaa (Bertino ja Catania, 2001).

Kuvan 43 mukainen rakenteeton esitystapa mahdollistaa erilaisen DTD:n omaavien XML-dokumenttien varastoinnin samaan tietokantatauluun.

Hybridi esitystapa tarkoittaa tapaa esittää sekä rakenteetonta että rakenteista dataa samassa XML-dokumentissa. Tyypillinen tällainen dokumentti on esimerkiksi kirjan rakenteen ja sisällön kuvaus. Sekalaista rakennetta käytettäessä tietokannanhallintajärjestelmältä vaaditaan kyselykielen laajentamista tietyillä XML-pohjaisilla valintaehdoilla.

5.2 Datan integrointi ja XML

Monitietokantajärjestelmien tarkoitus on saavuttaa jonkinasteista datan integrointia ja tietokantojen välistä toiminnallisuutta. Datamallien ja kaavojen heterogeenisyys tekevät datan integroinnin ja eri muodossa olevien tietokantojen välisen kommunikaation ja tiedon siirron hankalaksi. Standardoitu XML-kieli tekee datan esittämisen ja välittämisen Webin kautta erilaisista laitealustoista ja sovelluksista huolimatta mahdollisek-

si. Lisäksi XML-pohjainen tietomalli sopii eri tyyppisten järjestelmien yhteenliittämiseen ja mahdollistaa datan integroinnin (Li & al., 2003).

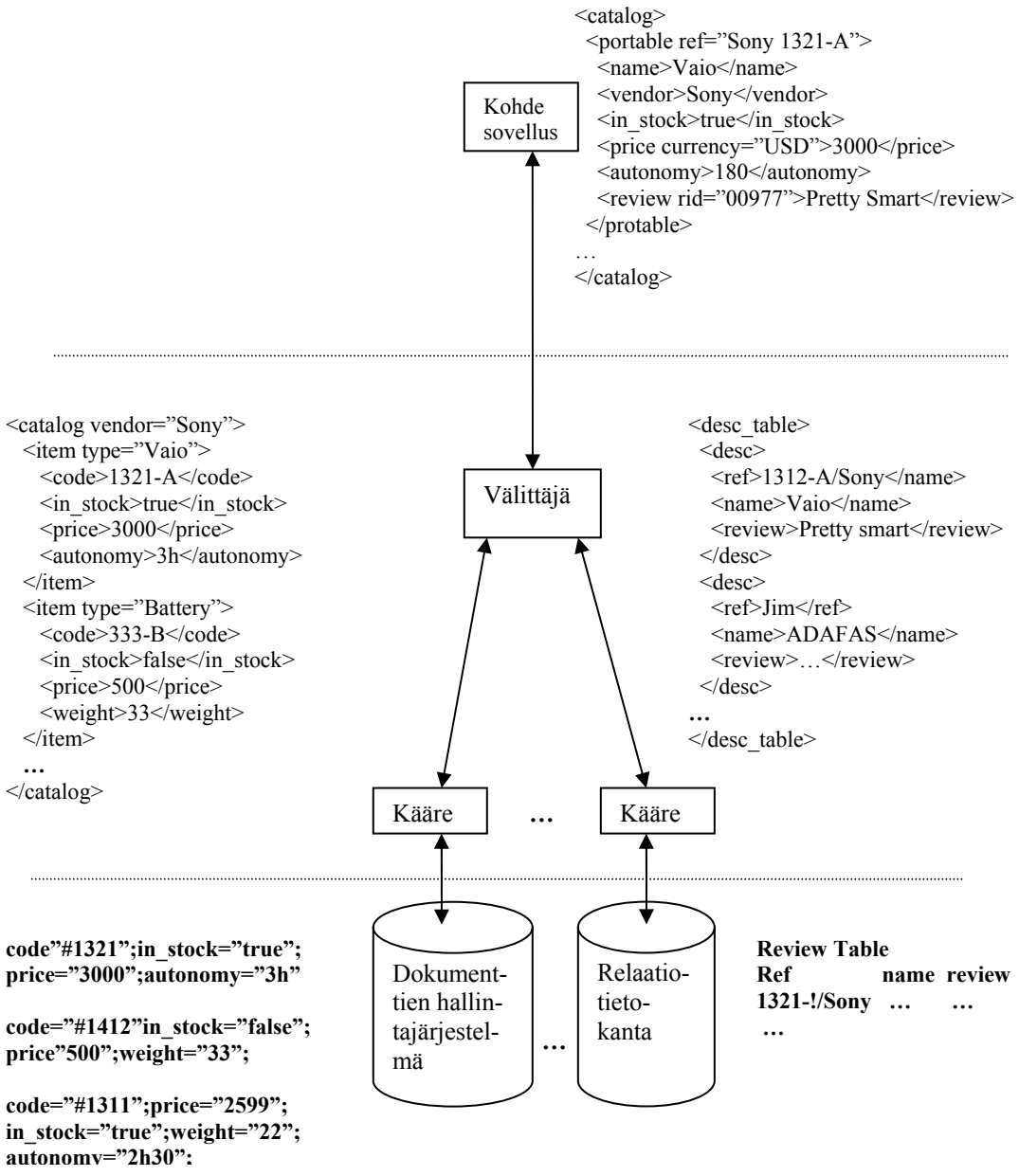
Ensink & al. (1999) käyttävät XML-pohjaista ratkaisua tiedon hankkimiseksi kolmesta erillisestä, itsenäisestä tietokannasta. Tietokannat sisältävät tietoa kirjakauppojen kirjoista. Kaikki kolme tietokantaa ovat sisäiseltä rakenteeltaan erilaisia. Tietotyypit, taulujen ja attribuuttien määrät eroavat toisistaan jokaisessa tietokannassa. Vaikka tietokannat ovat hyvin erilaisia, ne kuvaavat saman kirjaentiteetin, jolle voidaan määritellä tietty yhteinen attribuuttijoukko. Esimerkiksi kirjalla on ISBN, otsikko, julkaisija ja yksi tai useampi kirjoittaja, kuten kuvassa 44. Ratkaisuna on siis *yhteisten* attribuuttien kuvaaminen XML-kielellä.

```
<book>
        <isbn>…</isbn>
        <title>…</title>
        <authors>
                <a1>…</a1>
                <a2>…</a2>
                …
        </authors>
        …
</book>
```

Kuva 44. Kirjaentiteetti.

Kuvan 44 mallinne sisältää kirjaentiteetin määrittelyn. Mallinteen ensimmäisellä rivillä määritellään tagi <book>, joka osoittaa uuden tietueen alun. Tämän tagin sisään on määritelty muita tageja kuvaamaan kirjaentiteetin attribuutteja. Kirjan tekijät määritellään <authors> -tagilla ja tämän sisään on määritelty elementtejä, jotka muodostavat listan kirjan kirjoittajista.

XML-kieltä voidaan hyvin soveltaa myöskin välittäjä-arkkitehtuuriin kuvan 45 mukaisesti mallinteiden rakentamiseksi. Kuvan 45 ylin kerros esittää kohdesovellusta, joka käyttää välittäjää tiedonhakuun tietolähteistä. Keskimmäinen kerros esittää XML-pohjaista datan integrointijärjestelmää. Siihen kuuluvat välittäjä ja XML-pohjaiset kääreet. Alin kerros koostuu tietolähteistä. Tietolähteitä voivat olla esimerkiksi dokumenttienhallintajärjestelmä ja relaatiotietokanta.



Kuva 45. XML-pohjainen datan integrointi (Patel-Schneider ja Siméon, 2003).

Petrou & al. (1999) esittävät välittäjien ja kohdesovelluksen välisille pyynnöille ja vastauksille kuvien 46 ja 47 mukaiset DTD-määrittelyt.

```
<?XML version=”1.0”?>
<!DOCTYPE schema_response [
<!ELEMENT schema (transaction, database)>
<!ELEMENT transaction (#PCDATA)>
<!ELEMENT database (table+, relationship*)>
<!ATTLIST database name CDATA #REQUIRED>
<!ELEMENT table (field+)>
<!ATTLIST table name CDATA #REQUIRED
        alias CDATA #IMPLIED
        option CDATA #IMPLIED>
<!ELEMENT field EMPTY>
<!ATTLIST field name CDATA #REQUIRED
        alias CDATA #IMPLIED
        dataType (LONG | STRING | …) #REQUIRED
        maxLength CDATA #IMPLIED
        precision CDATA #IMPLIED
        position CDATA #IMPLIED
        defaultValue CDATA #IMPLIED
        nullable (YES | NO) “YES”>
<!ELEMENT relationship EMPTY>
<!ATTLIST relationship field1 CDATA #REQUIRED>
        field2 CDATA #REQUIRED
        cardinality CDATA
        mandatory (YES | NO) “YES”>
]>
```

Kuva 46. DTD:n määrittely välittäjän ja kohdesovelluksen väliselle pyynnölle.

```
<?XML version=”1.0”?>
<!DOCTYPE results_response [
<!ELEMENT results (transaction, code, data)>
<!ELEMENT transaction (#PCDATA)>
<!ELEMENT code EMPTY>
<!ATTLIST code value CDATA #REQUIRED>
<!ELEMENT data (record+)>
<!ELEMENT record (d+)>
<!ATTLIST record count CDATA #REQUIRED>
<!ELEMENT d (#PCDATA)>
]>
```

Kuva 47. DTD:n määrittely välittäjän ja kohdesovelluksen väliselle vastaukselle.

Datan integrointia ei XML-pohjaisessa ratkaisussakaan ole helppo suorittaa, mikäli semanttiset seikat eivät ole täysin ratkaistu. Ratkaisuksi Patel-Schneider ja Siméon (2003) tarjoavat kohdassa 4.3 tarkastellun RDF:n sekä XML-kaavan hyödyntämistä. XML-kaavaa datan integroimiseen käyttävät myös Shui ja Wong (2003). XML-kaava

ei rajoita DTD:n tavoin datan käsittelyä, vaan sallii muun muassa erilaisten tietotyyppien käytön ja mahdollistaa periytymisen (Sperberg-McQueen ja Thomson, 2004).

XML-kielen hyödyntämiseksi on toteutettu erilaisia rajapintoja, kuten Simple API for XML eli SAX ja Document Object Model eli DOM (Linthicum, 2001). Näistä rajapintoja hyödyntämällä XML-dokumentteja voidaan muuttaa toiseen muotoon tai muille kielille XSLT-kielen (eXtensible Stylesheet Language: Transformations) ja Xpath-kielen (XML Path Language) avulla (W3C, 2004).

6 YHTEENVETO

Yhdistetty tietokantajärjestelmä on kokoelma yhdistettyjä tietokantoja ja niiden käytön mahdollistavia tietokannan hallintajärjestelmiä. Yhdistettyyn tietokantajärjestelmään kuuluvien tietokantojen välille luodaan yhteys. Tietokannat voivat kommunikoida keskenään yhdistetyn tietokannan hallintajärjestelmän avulla. Yhdistetty tietokannan hallintajärjestelmä mahdollistaa tiedonvälityksen yhdistettyyn tietokantajärjestelmään kuuluvien tietokantojen kesken tietomallien erilaisuudesta huolimatta.

Yhdistetty tietokannan hallintajärjestelmä edellyttää arkkitehtuuria, mikä sisältää tietynlaisia komponentteja. Kaksi peruskomponenttia, prosessorit ja kaavat, ovat tärkeitä arkkitehtuurin määrittelyssä. Prosessorit suorittavat erilaisia toimintoja datalle ja komennoille. Kaavat koostuvat kaava-alkioista ja niiden välisistä suhteista. Kaavat kuvaavat käsitteellisiä ja loogisia tietorakenteita ja tietorakenteiden välisiä suhteita. Prosessorit sijaitsevat arkkitehtuurissa kaavojen välissä. Toimiakseen yhdistetty tietokantajärjestelmä tarvitsee tietokantajärjestelmän hoitajan, joka käsittelee kaavojen välisiä suhteita.

Yhdistetty tietokantajärjestelmä integroi sekä tietokantoja että tietokantojen sisältämää dataa. Yhdistetyn tietokannan hallintajärjestelmän rakentaminen ja ylläpito verrattuna muihin integrointimenetelmiin on työläs ja haastava prosessi. Yhdistettyihin tietokantajärjestelmiin liittyvä tutkimus ja kehitystyö on vähentynyt viime vuosina, osittain puuttuvien ohjelmistotyökalujen vuoksi, osittain menetelmän vaatiman työmäärän vuoksi. Muut tässä tutkielmassa esitellyt menetelmät soveltuvat yhdistettyä tietokantajärjestelmää paremmin tietokantojen ja datan integrointiin helppokäyttöisempinä ja halvempina metodeina.

Tietovarastointi on tiedon integrointiarkkitehtuuri, jossa useista itsenäisistä ja heterogeenisistä lähteistä kerätty tieto muodostaa yhden globaalin kaavan. Tietovarastoinnin tarkoitus on parantaa päätöksentekoa liiketoiminnassa. Päätöksentekoa varten tarvitaan historiatietoja. Historiatietojen saamiseksi, ja tietovaraston ajantasaisuuden vuok-

si, tietovarastoa päivitetään aika ajoin. Käyttäjä voi kohdistaa tietovarastoon kyselyitä, kuten mihin tahansa tietokantaan.
Tietovaraston tiedot kerätään tietolähteistä poimijoiden avulla. Poimijat välittävät kerätyn tiedon yhdistäjälle. Yhdistäjä integroi poimijoilta saamansa tiedon ja lataa datan tietovarastoon. Ennen datan lataamista tietovarastoon poimittu tieto on puhdistettava. Puhdistus tapahtuu tavallisesti erillisellä muunnosalueella.

Tietovaraston arkkitehtuurina on usein moniulotteinen malli, joka tukee päätöksenteon analysointitekniikoita. Analysointitekniikoita ovat erilaiset tilastolliset menetelmät. Tietovarastointi mahdollistaa myös tiettyjen suorituskykyä parantavien tekniikoiden, kuten indeksoinnin, käytön.

Tietovarastointitekniikkaa käytetään laajalti yrityksissä tietokantojen integrointikeinona. Tekniikan käyttöönotto on helppoa ja edullista olemassa olevien työkalujen vuoksi. Halutunlaisen tietovaraston rakentaminen yrityksen sisällä on myös mahdollista tekniikan soveltamisen helppouden ja käyttökelpoisuuden takia. Tietovarastointiin liittyvät läheisesti välittäjien ja kääreiden käyttö datan integroinnin ja muuntamisen apuna.

Välittäjät ovat ohjelmistomoduuleja, jotka välittävät dataa sovellusten ja tietolähteiden välillä. Välittäjät muuntavat ja integroivat dataa tiedonvälityksen mahdollistamiseksi ja tiedon ylimäärän karsimiseksi. Välittäjät eivät varastoi dataa, mutta ne voivat säilyttää välituloksia. Välittäjien käytetään esimerkiksi tietovarastosovelluksissa. Kääreet mahdollistavat datan saannin heterogeenisista tietolähteistä. Kääre on ohjelma, joka kääntää käyttäjän tietolähteille esittämät kyselyt tietolähteen ymmärtämään muotoon. Käyttäjän kysely tulee kääreelle tavallisesti välittäjän kautta. Jokaista tietolähdettä kohden rakennetaan kääre, joten tietolähteiden data voi olla missä muodossa tahansa.

Välittäjät integroivat tietokannoista välitettävää dataa. Välittäjiin on mahdollista ohjelmoida tilastollisia menetelmiä hyödyntävää softaa. Välittäjien käytettävyyttä parantavat yleiskäyttöisyys sekä työkaluihin liitettävä jonkinasteinen mahdollisuus käyttäjän omiin määrityksiin. Kääreet ovat käyttökelpoisia käyttäjän kyselyiden ja tietokannoista välitettävän datan muuntamisessa.

XML on tietojen kirjoittamisen avoin standardi, jonka avulla tietoja voidaan jakaa ohjelmistojen ja järjestelmien avulla. XML eroaa muista merkkauskielistä piirteidensä vuoksi. XML mahdollistaa muun muassa käyttäjän määrittelemät tagit, jotka sallivat datan ja datan kuvauksen esittämisen samassa dokumentissa.

XML-dokumentit jaetaan datakeskeisiin ja dokumenttikeskeisiin dokumentteihin. Datakeskeiset dokumentit ovat säännöllisiä rakenteeltaan ja homogeenisiä sisällöltään. Dokumenttikeskeinen XML-dokumentti sisältää suurimmaksi osaksi rakenteetonta dataa. XML-dokumenttien esittämiseen on kaksi esitystapaa: rakenteeton ja hybridi esitystapa. Rakenteeton esitystapa käsittelee XML-dataa yhtenä objektina. Hybridi esitystapa tarkoittaa tapaa esittää rakenteetonta ja rakenteista dataa samassa XML-dokumentissa.

Standardoitu XML-kieli tekee datan esittämisen ja välittämisen Webin kautta erilaisista laitealustoista ja sovelluksista huolimatta mahdolliseksi. Lisäksi XML-pohjainen tietomalli sopii eri tyyppisten järjestelmien yhteenliittämiseen ja mahdollistaa datan integroinnin. XML-kieli tietokantojen ja datan integrointikeinona on edullinen ja käyttökelpoinen. XML-kieli on tietovarastoinnin ja välittäjätekniikan ohella yhdistettyjä tietokantajärjestelmiä helpompi tapa integroida tietokantoja.

VIITELUETTELO

Batini C., Lenzerini M.: A Comparative Analysis of Methodologies for Database Schema Integration, *ACM Computing Surveys,* 18(4), 1986, 324-326.

Bertino E., Catania B.: Integrating XML and Databases, *IEEE Internet Computing,* Jul/Aug, 2001, 84.

Bray T.: What is RDF, *O'Reilly Open Source Convention,* 26-30th July, Portland, Oregon, USA, 2001, http://www.xml.com/ (26.4.2004).

Connolly T.M., Begg C.E.: *Database Systems: A Practical Approach to Design, Implementation, and Management,* Third Edition, Pearson Education Limited, Edinburgh Gate, Harlow, Essex CM20 2JE, England, 2002.

Critchlow T., Ganesh M., Musick R.: Meta-data based mediator generation, Cooperative Information Systems. *Proceedings, 3rd IFGIS International Conference,* NY, USA, 20-22 Aug. 1998, 168-176.

Dean M., Schreiber G. (Eds.): *OWL Ontology Language Reference,* W3C Proposed Recommendation, 9th Dec. 2003, http://www.daml.org/ (27.4.2004).

Deitel H.M., Deitel P.J., Nieto T.R., Lin T.M., Sadhu P.: *XML: how to program,* Prentice-Hall, Inc. Upper Saddle River, New Jersey 07458, USA, 2001.

Elmasri R., Navathe S.B.: *Fundamentals of Database Systems,* Third Edition, Buschlen/ Mowatt Fine Arts Ltd., Main Floor, 1445 West Georgia Street, Vancouver, Canada, 2000.

Ensink B., Haveman K., Shrestha M., Schavey T.: XML based Adaptation of the Composite Approach for Database Integration, *Proceedings of the 37th Annual ACM Southeast Regional Conference,* ACM Press, NY, USA, 1999, 15-18.

El-Masri R., Wiederhold G.: Data Model Integration Using the Structural Model, *Proc. ACM Sigmod Conf.,* ACM Press, New York, 1976, 191-202.

Furlow G.: The case for building a data warehouse, *IT Professional,* 3(4), Jul/Aug 2001, 31-34.

Garcia-Molina H., Papakonstantinou Y., Quass D., Rajaraman A., Sagiv Y., Ullman J., Widom J.: *The TSIMMIS Approach to Mediation: Data Models and Languages (Extended Abstract),* Stanford University, Stanford CA, 94305, USA, 1997.

Garcia-Molina H., Ullman J.D., Widow J.: *Database System Implementation,* Department of Computer Science, Stanford University, Prentice-Hall, Inc. Upper Saddle River, New Jersey 07458, USA, 2000, 595-612.

Ghidini C., Serafini L.: Model Theoretic Semantics for Information Integration, *AIMSA '98* (Ed. F. Giunchiglia), LNAI 1480, 1998, 267-280.

Hearst M.A., Levy A.Y., Knoblock C., Minton S., Cohen W.: Information integration, *IEEE Intelligent Systems,* 13(5), Sept/Oct 1998, 12-24.

Hobbs L., Hillson S.: *Oracle 8i Data Warehousing,* Digital Press, Butterworth-Heinemann, 225 Wildwood Avenue, Woburn, MA 01801-2041, USA, 2000, 15-19.

Hovi A., Huotari J., Lahdenmäki T.: *Tietokantojen suunnittelu ja indeksointi,* Docendo Finland Oy, Jyväskylä, 2003.

Inmon W.H.: *Building the Data Warehouse,* Second Edition, Wiley Computer Publishing, John Wiley & Sons, Inc. USA, 1996.

Kamel M.N., Kamel N.N.: The Federated Database Management System: An Architecture of Distributed System for the 90's, *Distributed Computing Systems Proceedings, Second IEEE Workshop on Future Trends of,* 30 Sep. - 2 Oct. 1990, 346-352.

Kedad Z., Métais E.: *Dealing with Semantic Heterogeneity during Data Integration,* ER'99 (Ed. J. Akoka et al.) LNCS 1728, Springer-Verlag Berlin Heidelberg, 1999, 325-339.

Laux M.: *DBAccessor: A JDBC Wrapper Pagggage,* June, 2002. Http://java. sun. com/developer/technicalArticles/Database/dbaccessor/

Lear A.C.: XML Seen as Integral to Application Integration, *It Pro,* Sept/Oct, 1999, 12.

Li R., Lu Z., Xiao W., Li B., Wu W.: Schema Mapping for Interoperability in XML-Based Systems, *IEEE Proceedings of the 14th International Workshop on Database and Expert Systems Application (DEXA'03),* 2003.

Linthicum D.S.: *B2B Application Integration,* Addison-Wesley, Boston, 2001.

Liu J., Vincent M.: An architecture for data warehouse systems, *TENCON '98, IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control,* New Delhi, India, 1998, 107-110.

Manola F., Miller E. (Eds.): *RDF Primer,* W3C Recommendation 10 February, 2004, 4-5. Http://www.w3org/TR/2004/REC-rdf-primer-20040210/

Parent C., Spaccapietra S.: Issues and Approaches of Database Integration, *CACM* 41(5), 1998, 166-178.

Patel-Schneider P.F., Siméon J.: The Yin/Yang Web: A Unified Model for XML Syntax and RDF Semantics, *IEEE Transactions on Knowledge and Data Engineering,* 15(4) Jul/Aug, 2003, 797-812.

Petrou C., Hadjiefthymiades S., Martakos D.: An XML-based, 3-tier scheme for integrating heterogeneous information sources to the WWW, *Database And Expert Sys-*

tems Applications, Proceedings. 10th International Workshop on, 1-3 Sept. 1999, 706-710.

Powers S.: *Practical RDF,* O'Reilly, Cambridge, 2003.

Priebe T., Pernul G.: Ontology-based Integration of OLAP and Information Retrieval, *IEEE Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA '03),* 2003.

Renner A.: XML Data and Object Databases: The Perfect Couple?, *IEEE Proceedings, 17th International Conference on,* 2-6 April, 2001, 143-148.

Roy J., Ramanujan A.: XML schema language: taking XML to the next level, *IT Professional,* 3(2), Mar/Apr, 2001, 38.

Sheth A.P., Larson J.A.: Federated Database System for Managing Distributed, Heterogenous and Autonomous Databases, *ACM Computing Surveys,* 22(3), 1990, 183-236.

Shui W.M., Wong R.K.: Application of XML Shema and Active Rules System in Management and Integration of Heterogeneous Biological Data, *Proceedings of the 3rd IEEE Symposium on Bioinformatics and BioEngineering (BIBE'03),* 2003.

Sperberg-McQueen C.M., Thompson H.: *XML Schema,* World Wide Web Consortium, http://www.w3.org/XML/Schema (27.4. 2004).

Sung H.H., Sang C.P.: Data Modeling for Improving Performance of Data Mart, *International Conference on Engineering and Technology Management IEMC'98. Proceedings of the IEMC'98,* San Juan, Puerto Rico, 1998, 436-441.

Theodoratos D., Sellis T.:Dynamic Data Warehouse Design*, DaWaK'99,* (Ed. Mukesh Mohania and A Min Tjoa) LNCS 1676, 1999, 1-10.

Travis G.: Java technology: An easy JDBC wrapper, *IBM developerWorks,* Aug. 2001, 1-5, http://www-106.ibm.com/developerworks/java/libryary/j-jdbcwrap/ (28. 2. 2004).

Verzulli J.: Using the Jena API to Process RDF, *O'Reilly Open Source Convention,* 26-30th July, Portland, Oregon, USA, 2001, http://www.xml.com/ (28.2.2004).

W3C, *The Extensible Stylesheet Language Family (XSL),* World Wide Web Consortium, http://www.w3c.org/style/XSL/(24.4.2004).

White C., Quin L., Burman L.: *Mastering XML,* SYBEX inc. 1151 Marina Village Parkway, Alameda, CA 94501, 2001.

Wiederhold G.: Mediators in the architechture of future information systems, *Computer,* 25(3), 1992, 38-49.

Wiederhold G., Genesereth M.: The Conceptual Basis for Mediation Services, *IEEE Expert,* 12(5), 1997, 38-47.

Wilkinson J.H., Morgan O.F.: Wrapper files – the key to network technology, *International Broadcasting Convention '97,* 12-16 Sep. (ed. Sony Broadcast & Professional Europe, UK), Amsterdam, Netherlands, 1997, 374-379.