Aarne Kauppinen

TIEDON POIMINTA, YHTENÄISTÄMINEN JA LATAAMINEN TIETOVARASTOON

Tietojärjestelmätieteen
Pro gradu –tutkielma
09.08.2000

Jyväskylän Yliopisto
Tietojenkäsittelytieteiden laitos
Jyväskylä

TIIVISTELMÄ

Kauppinen, Aarne Johannes
Tiedon poiminta, yhtenäistäminen ja lataaminen tietovarastoon / Aarne Kauppinen.
Jyväskylä: Jyväskylän yliopisto, 2000.
92 s.
Tutkielma

Tutkielman tarkoituksena on perehtyä tietovaraston rakentamiseen ja siihen liittyviin ongelmiin. Tässä tutkielmassa keskitytään tietojen keräämiseen operatiivisista ja muista ulkopuolisista lähdejärjestelmistä, näiden tietojen vaatimaan esikäsittelyyn ja siirtämiseen tietovarastoon. Tutkimuksen tavoitteena on selvittää, miten tietovarastosovellus rakennetaan ja millaiset taulurakenteet siinä tarvitaan. Suuria tietomassoja käsitellessä halutaan usein näyttää summattuja tietoja. Tämän vuoksi summauksen tekeminen tietojen haun yhteydessä on välttämätöntä. Vastaavasti tietoihin halutaan päästä käsiksi useista eri näkökulmista. Nämä tekijät yhdessä yleensä aiheuttavat suuria ongelmia tietojen tallentamisten, poiston ja hakujen tehokkuuteen.

Tutkimustyö on tehty perehtymällä tietovarastoinnista julkaistuun kirjallisuuteen ja materiaaliin. Lisäksi käytännön kokemuksia tietovarastosovellusten rakentamisesta ja ylläpidosta on saatu erään suuren telealan yrityksen tietohallinnosta, joka on rakentanut tietovarastosovellutuksia lähinnä omaan asiakashallintaansa.

Tietovarasto koostuu fakta-, dimensio- ja summatauluista. Dimensiotaulut ovat näkökulmia faktatauluun. Dimensiotaulut eivät sisällä varsinaista dataa vaan viiteavaimia, joilla faktataulut tiedot voidaan saada esille. Tärkein tietovaraston dimensio on aika. Aika määrää tietovaraston rakeisuuden eli sen, millä ajallisella tarkkuudella tieto tallennetaan faktatauluun. Summataulut sisältävät valmiiksi summattua tietoa nopeuttamaan tiedon hakua tietovarastosta.

Kuutioita tarvitaan tiedon moniulotteiseen hakemiseen. Kuutiot pitää rakentaa tietovaraston taulujen ulkopuolelle erillisellä ajolla, minkä jälkeen siihen voi pureutua erillisellä raportointityökalulla. Tiedon louhinnalla tarkoitetaan monimutkaista tiedon hakemista tietovarastosta annettujen kriteerien perusteella. Tiedon haku hakee syitä ja lainalaisuuksia. Jotta tietovaraston käyttö olisi mahdollista, täytyy käyttäjien tietää, mitä tietovarasto sisältää. Tietoa tietovaraston sisällöstä nimitetään metadataksi. Se on edellytys tietovaraston täysipainoiselle käytölle.

Tietovarastointi käsittää kolme perustehtävää: tiedon hankita, tallennus ja esille saanti. Tietovaraston tiedon hankinnan katsotaan käsittävän tiedon poiminnan

lähdejärjestelmistä, yhtenäistämisen, siirron ja latauksen tietovarastoon. Hankittava tieto täytyy tunnistaa, sen tietosisältö pitää arvioida ja haluttu tieto pitää poimia. Poiminta voi olla muuttumatonta, heti tapahtuvaa tai viivästettyä poimintaa. Tiedon yhtenäistäminen on monimutkainen ja vaativa tehtävä, joka sisältää tiedon muunnoksia, yhtenäistämistä ja usean lähtöjärjestelmän tietojen yhdistämistä, summausta ja virheellisen tiedon hallintaa. Poimitut tiedot pitää ladata tietovarastoon ensin alkulatauksella ja sitten tietovaraston käytön aikaisilla jaksottaisilla latauksilla.

Sovelluksien, joiden tulee tallentaa päivittäin miljoonia rivejä tietoa, ongelmaksi tulee hyvin pian suuren tietomäärän hallinta ja hakuaikojen kasvaminen. Tietovaraston rakentaminen kannattaa aloittaa yhdestä osa-alueesta, jolla kerätään käytännön kokemuksia tietorakenteiden suunnittelusta, taulujen ja indeksien perustamisesta sekä latausprosessin toteuttamisesta. Tietovaraston rakentaminen on jatkuva prosessi, mikä toivottavasti helpottaa sovellusten ylläpitoa ja muutosten toteuttamista.

AVAINSANAT: tietovarastointi, data warehousing, latausprosessi

SISÄLLYSLUETTELO

1 Johdanto 1
2 Tietovarastointi 4
2.1 Tietovarastoinnin erityispiirteitä 4
2.2 Tietovaraston taulut 6
2.2.1 Taulurakenteet 7
2.2.2 Faktataulut 11
2.2.3 Dimensiotaulut 14
2.2.4 Summataulut 15
2.3 Tiedon moniulotteinen käsittely 18
2.3.1 Kuutiot 20
2.3.2 Data mining 22
2.4 Metadata 22
2.5 Tekninen ratkaisu 24
2.5.1 Rinnakkaiskäsittely 25
2.5.2 Ositus 26
3 Tiedon hankinta 29
3.1 Sisäiset ja ulkoiset lähdejärjestelmät 31
3.2 Tietovarastoon tallennettavan tiedon kolme tyyppiä 33
3.3 Tiedon poiminta 36
3.3.1 Tiedon tunnistaminen 37
3.3.2 Tietosisällön arviointi 38
3.3.3 Tiedon poimimisen operatiivisista järjestelmistä 39
4 Tiedon yhtenäistäminen 50
4.1 Yhtenäinen tieto 50
4.2 Laatutavoitteet 52
4.3 Tietoihin tehtävät muunnokset 53
4.3.1 Merkkimuunnokset 54
4.3.2 Käyttöjärjestelmämuunnokset 54

4.3.3 Tiedon sisäiset muunnokset 55
4.3.4 Tiedon yhtenäistäminen 56
4.3.5 Useiden eri lähtötietojen yhdistäminen 57
4.3.6 Summaus ja yhdistely 57
4.4 Virheellisen tiedon hallinta 58
4.5 Ristiviittausten tarkastaminen 59
5 Tiedon lataus tietovarastoon 61
5.1 Alkulataus 63
5.2 Jaksottaiset lataukset 66
5.3 Tietovaraston päivittäminen 68
6 Yhteenveto 71
Tutkimuksen lähteet 75
Liitteet 78

1 JOHDANTO

Päätöksentekijän kannalta operatiivisten tietojärjestelmien suuret tietomassat eivät ole vieläkään oikeassa muodossa tai oikealla tavalla käsiteltävissä. Asiaa on pyritty ratkaisemaan monilla eri tavoilla. 1970-luvun alun johdon tietojärjestelmien (*MIS, Management Information System*) ajatuksena oli rakentaa joukko toisiinsa kytkeytyviä tietojärjestelmiä, jotka sisältävät yrityksen kaiken tiedon. Rakennusyrityksistä luovuttiin osin tekniikan kypsymättömyyden, osin tällaisen ratkaisun mutkikkuuden vuoksi. Saman vuosikymmenen lopulla nousivat esiin erilaiset päätöksenteon tukijärjestelmät (*DSS, Decision Support System*). Niihin kehitettiin voimakkaita yrityksen mallinnusominaisuuksia ja eräissä tapauksissa tulokset liiketoiminnan ohjauksessa olivat hyviä. Ongelmaksi muodostui tällaisten ratkaisujen erillisyys yrityksen päivittäistä toimintaa palvelevista tietojärjestelmistä. 1980-luvun puolivälissä rakennettiin ns. Info Centereitä, joiden tarkoituksena oli muokata operatiivisten järjestelmien tiedot sellaiseen muotoon, että käyttäjät voivat itse laatia haluamiaan raportteja ja analyyseja. Monissa tapauksissa onnistuttiin tekemään tähän tarvittavia infokantoja, mutta käyttäjien omien analyysien pyörittäminen keskustietokoneilla tuli kalliiksi. Tuon ajan mikrot eivät puolestaan riittäneet kovinkaan vaativaan käsittelyyn.

Yritysten tietojärjestelmissä on erittäin paljon tietoa, jota ei pystytä hyödyntämään päätöksenteossa tai asiantuntijatyössä. Asiaa on pyritty ratkaisemaan monilla eri tavoilla, uusimpana ratkaisutapana tietovarastot (*Data Warehouse*). Tietovarastointi nousi terminä esiin 1990-luvun alussa, kun Bill Inmon konsulttina sekä IBM ja Digital teknologiavalmistajina, julkistivat ratkaisumallin melkein samaan aikaan. Bill Inmon on kuvannut tietovarastoa ennen kaikkea ratkaisuna yrityksen tiedonhallinnan ongelmiin: miten yhdistää eritasoisissa ja eri-ikäisissä tietojärjestelmissä oleva epäyhtenäinen tieto järkevästi asiantuntijoiden ja päätöksentekijöiden käyttöön. IBM julkisti oman näkemyksenä, Information Warehouse Framework, vuonna 1991. Tarkoituksena oli tuottaa kerätystä tiedosta in-

formaatiota. Samankaltaiset ajatukset toistuvat myös Digitalin vuonna 1992 julkaisemassa mallissa Data Warehouse Strategy.

Tässä tutkielmassa tarkastellaan tietojen keräämistä operatiivisista ja muista ulkopuolisista lähdejärjestelmistä, näiden tietojen vaatimaa esikäsittelyä ja siirtämistä tietovarastoon. Varsinkin tietojen yhtenäistäminen on kaikkien ongelmallisin ja eniten resursseja vievä osuus. Tietojen yhtenäistämisellä tarkoitetaan tietoihin tehtäviä muunnoksia ja korjauksia, jotta tietovarastoon siirrettävä tieto olisi virheetöntä, yhdenmukaista ja ristiriidatonta. Matka- ja dataviestinnän sekä kiinteän verkon puhelinpalveluja tarjoavan yrityksen, Soneran tietovarastosovelluksia rakentavan tietohallinnon kokemuksien mukaan, tietojen yhtenäistäminen vie noin 70-80% resursseista, joita tarvitaan tiedon on tallentamistoimenpiteisiin tietovarastotietokantaan (ks. myös 2000, http://www.infomanager.fi/ datawa-re.htm >).

Tutkimuksen tavoitteena on selvittää ongelmat, joita kohdataan tietojen poiminnassa, yhtenäistämisessä ja lataamisessa tietovarastoon. Selvitystyö tehdään perehtymällä tietovarastoinnista julkaistuun kirjallisuuteen ja valmiisiin tietovarastosovellutuksiin. Tarkoituksena on pystyä ennakoimaan ongelmat ja valmistautumaan niihin kunnolla.

Tutkimustyön tuloksia pyritään soveltamaan käytännön sovellukseen ja poistamaan nykyiset ongelmakohdat: tietojen lataus on hidasta, tietojen hakuajat liian pitkiä ja vanhentuneen tiedon poisto on raskasta ja koneresursseja vievää.

Tämä tutkielma painottuu tietovarastoinnin ongelmakentän tutkimuksessa tiedon poimintaan, yhtenäistämiseen ja lataamiseen. Tutkielman luvussa 2 tarkastellaan tietovarastoinnin perusperiaatteita. Tässä yhteydessä käydään läpi tietovarastossa käytettävät taulurakenteet. Tietovarastossa käytettävät tietomallit ovat tähti- ja lumihiutalemalli, joita vastaavat relaatiotietokannassa fakta-, dimensio- ja summataulut. Luvussa 3 tutustutaan tietovaraston tietojen hankintaan. Lähdejärjestelmät voivat olla organisaation sisäisiä tai ulkoisia. Tietojen hankinta voidaan

jakaa tietojen poimintaan lähdejärjestelmistä, yhtenäistämiseen ja tietovarasto-tallennukseen. Luvussa 4 perehdytään tietojen yhtenäistämiseen, ja lopuksi luvussa 5 kuvataan tietojen erilaiset tallentamistavat.

2 TIETOVARASTOINTI

Tietovarastointi (*data warehousing*) kerää yrityksen avainprosesseihin liittyvää luotettavaa tietoa organisaation sisä- ja ulkopuolisista lähdejärjestelmistä, yhdistelee, yhdenmukaistaa ja siivoaa sen. Tämän jälkeen tieto on enemmän kuin vain kokoelma erillisiä tietoja irrallisista asioista; tiedot on yhdistetty sovittujen liiketoimintaprosessissa käytettyjen käsitteiden ja termien mukaiseksi kokonaisuudeksi. Kun aikaisemmin useissa erillisissä järjestelmissä sijainnut tieto on yhdistetty hallittavaksi kokonaisuudeksi, tieto on tällöin muutettu tietämykseksi ja on hyväksikäytettävissä päätöksenteon tukena organisaation eri tasoilla.

2.1 Tietovarastoinnin erityispiirteitä

Tietovarastossa säilytettävä tieto tallennetaan yleensä tietokantaan; osa tiedoista saattaa olla tiedostoina levyllä. Tällaisia tietoja ovat esimerkiksi tiivistetty vanha ja ei-aktiivinen tieto ja kuvat, joihin on vain linkki tietokantatietueesta. Tietovarastossa käytettävä tietokanta on yleensä relaatiotietokanta. Oliokantoja kehitetään ja ne yleistyvät tietovarastokäytössäkin. Inmon (1997b, s. 44) toteaa, että relaatiotietokannan ominaisuuksia yritetään parantaa kahdella eri tavalla. Koska tietovarastot ovat normaalisti pelkästään kyselykäytössä, tietokannanhallintajärjestelmien ei tarvitse hallita tietueiden lukitukseen ja samanaikaiseen päivitykseen liittyviä ongelmia. Muuttamalla tietokanta vain lukukäyttöön tarkoitetuksi voidaan eliminoida lokien kirjoitus. Normaalisti lokeja tarvitaan päivitettävien tietojen talletukseen siihen saakka, kunnes tiedot on tallennettu fyysisesti tietokantaan (Ault 1998, s.6). Kun normaalissa kyselykäytössä tietueiden lukitusta ei tarvita, tietokannan käsittely nopeutuu. Tietovaraston latauksen aikana voidaan lukituksia kuitenkin joutua käyttämään, sillä tietokannan eheys on varmistettava

useiden samanaikaisten latausprosessien, ja mahdollisesti samanaikaisten käyttäjien kyselyiden oikeiden tulosten varmistamiseksi. Toinen tapa parantaa tehokkuutta on käyttää tähtikaavio (*star schema*) -tietorakennetta.

Tietovarastossa oleva tieto on kohdesuuntautunutta (*subject oriented*), yhdistettyä (*integrated*), pysyvää (*non volatile*) ja aikaan sidottua (*timevariant*) (Inmon, 1996, s.33, Data Warehousing fundamentals Volume 1-3, 1996, s. 1-17). Tietovarastossa säilytetään tietoa pitkältä ajalta pääasiassa vain lukukäyttöön. Periaatteena on se, että tietoa ei päivitetä vaan ainoastaan lisätään säännöllisillä latauskerroilla yleensä enintään kerran vuorokaudessa. Tietovaraston sisältämä tieto on dokumentoitu metadataan, joka sisältää tietoa tietovaraston sisältämästä tiedosta ja sen käyttämistä käsittelysäännöistä.

Tietovarasto sisältää denormalisoitua tietoa: pääperiaatteena ei ole pyrkiä minimoimaan tiedon toistoa, vaan tehostamaan tiedon hakua. Tiedon haun nopeuttamiseksi tietovarastoon tallennetaan osa tiedoista valmiiksi summattuina useimmiten erillisiin summatauluihin.

Rob Mattisonin (1996, s.10) mukaan tietovarastointijärjestelmän rakentaminen jakautuu kolmeen osaan: tiedon hankinta, talletus ja saanti. Näistä työläin on tiedon hankinta, ja siksi tietovaraston suunnittelussa katsotaankin olevan kaksi tärkeää osaa: liittymien suunnittelu, joista tieto hankitaan ja toisaalta itse tietovaraston suunnittelu (Inmon 1999, s.73).

Tietovaraston käyttäjän kannalta tärkeintä on se, että tietovarastossa oleva tieto on oikeaa, se on helppo saada esille tarvittavassa muodossa ja haku on suhteellisen nopeaa verrattuna haettavan tiedon määrään. Tällöin tietovaraston käyttötavat voivat vaihdella etukäteen muotoilluista ja määräajoin ajetuista raporteista aina joustaviin tosiaikaisiin tiedon hakuihin, joissa pureudutaan haettavaan aiheeseen vaihtelevista näkökulmista. Suurin etu tiedon monipuolisista hakumahdollisuuksista on kuitenkin se, että tietoa voidaan hakea dynaamisesti useasta eri näkökulmasta. Esimerkiksi yrityksen myyntitietoihin voidaan päästä käsiksi myynti-

alueen, myyntimiehen, tuoteryhmän ja lopuksi yksittäisen tuotteen suunnasta halutusta ajanjaksosta. Toisaalta joku toinen käyttäjä voi haluta katsella tietoja tuoteryhmittäin ja myyntialueittain jakautuneena tiettyyn ajanjaksoon kiinnitettynä.

Tietovaraston käyttäjät voidaan jakaa karkeasti kolmeen eri tyyppiin: rajoittamattomat tiedon ad hoc -käyttäjät tiedon louhintatarkoituksiin, rajoitetun käytön ad hoc -käyttäjät ja valmiiden tietojärjestelmien avulla tietovarastoa käyttäviin käyttäjiin (Inmon, 1997a, s.297).

Hovin (1997, s.106) mukaan tietovaraston rakentaminen on jatkuva prosessi, joka ei pääty koskaan. Tietovarasto elää organisaation ja sen ympäristön muuttuessa; tietovarastoon tulee aina uusia tietotarpeita. Olemassa olevia operatiivisia järjestelmiä kehitetään ja niistä halutaan aina uusia tietoja myös tietovarastoon. Hovin mukaan tietovarastoon halutaan aikaa myöten aina uusia tietoja myös muista ulkopuolisista järjestelmistä. Tietovarastojärjestelmän tietomäärän kasvaessa tulee jatkuvasti tarpeita kehittää järjestelmän suorituskykyä ja parantaa vasteaikaa lisäämällä summatauluja ja kehittämällä jo olemassa olevia. Indeksointia ja partitiointia on kehitettävä, ja järjestelmää on tarkkailtava jatkuvasti, jotta levytila riittäisi ja vanhat tiedot siirrettäisiin historia-arkistoon.

2.2 Tietovaraston taulut

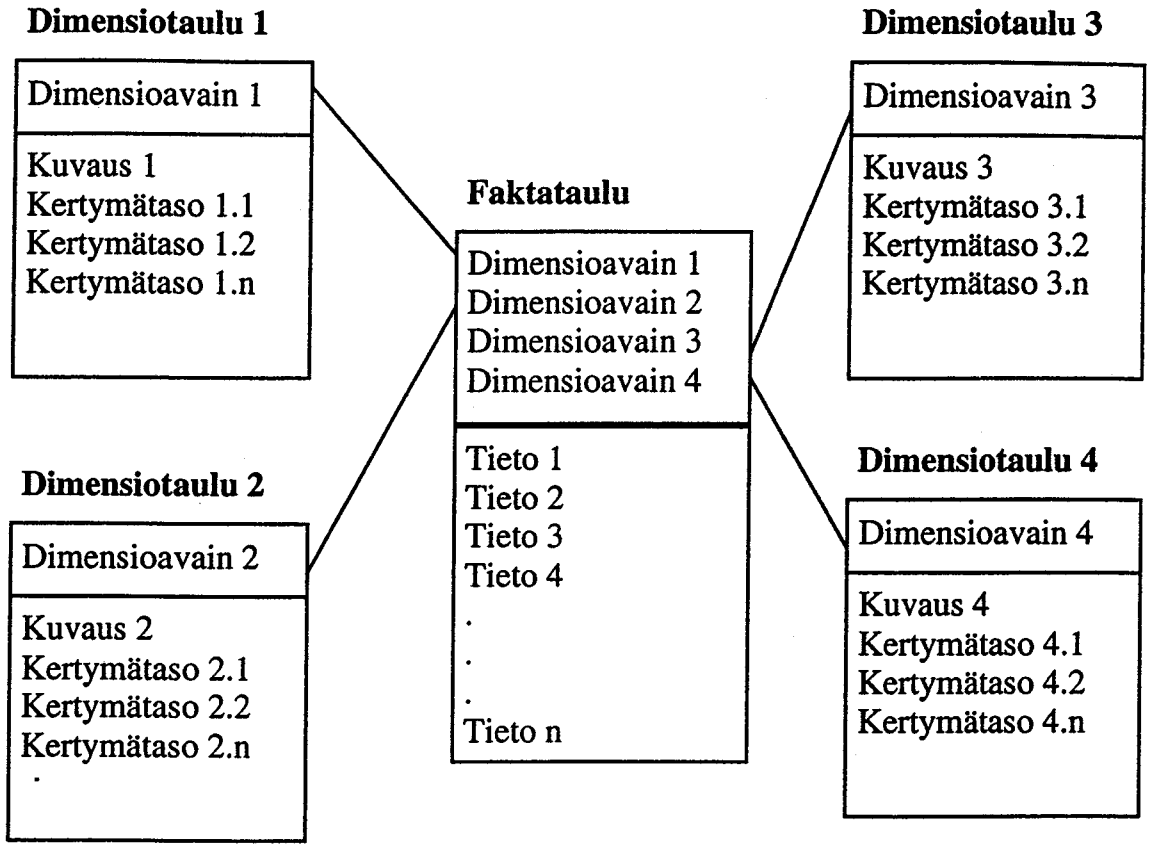
Tietovaraston tietojen haku perustuu usein hyvin monien tietojen yhdistelyyn ja monimuotoiseen hakuun. Jos tietokantatauluina käytettäisiin samoja rakenteita kuin operatiivisissa järjestelmissä, hakuprosessissa jouduttaisiin yhdistelemään lukemattomien taulujen tietoja, jotta tarvittava tulos saataisiin aikaan. Lisäksi operatiivisissa järjestelmissä on pyrkimyksenä nopeuttaa tietojen päivitystä, jolloin tiedon toisto ei ole suotavaa. Tietovarastotaulujen suunnittelussa yhdisteltävien taulujen lukumäärää halutaan minimoida, eikä normalisointia ei tarvitse vie-

dä samalle tasolle kuin operatiivisien järjestelmien taulurakenteissa. Tietovaraston denormalisoidut taulutyypit on tähtimalli (*starschema*) ja siitä johdettu normalisoitu lumihiutalemalli (*snowflake*). Näiden välimuoto on tähtihiutalemalli (*starflake shema)* ja se sisältää osia molemmista malleista. Tietovaraston suuri tietomassa on tallennettu faktatauluihin ja linkit siihen johdetaan dimensiotaulujen kautta. Faktataulu on yhteen tai useampiin tekijöihin (*dimensio*) kytketty suureita (*fakta*) sisältävä taulu (2000, http://www.cs.Helsinki.FI/u/laine/ relaatiosanasto/). Summataulut sisältävät etukäteen valmiiksi summattua tietoa nopeuttamaan tietojen hakua, sillä useimmat haut kohdistuvat jollain tavalla summattuun tietoon eivätkä yksittäisiin tietoalkioihin.

2.2.1 Taulurakenteet

Tietovaraston tietorakenteena tähtimalli muodostuu sekä fakta- että dimensiotauluista. Faktataulun voidaan kuvata asettuvan kaavion keskelle ja dimensiotaulut sen ympärille. Faktatauluun tallennetaan kaikki summattavissa olevat tiedot eli ne tiedot, joita tietorakenteen mukaan aiotaan hakea tietovarastosta. Faktataulu sisältää samalla myös viiteavaimet dimensiotauluihin. Faktataulu on kooltaan suurin taulu, sillä se sisältää periaatteessa yksin kaiken haettavan tiedon. Faktataulun tietoja ei yleensä koskaan muuteta, vaan tietoa aina lisätään (Anahory, 1997, s.43). Kuviossa 1 esitetään fakta- ja dimensiotaulujen suhteita.

Puhtaan tähtimallin mukainen tietorakenne toistaa dimensiotaulun tietoja. Toiston estämiseksi ja tilavarauksen pienentämiseksi on kehitetty tähtihiutalemalli (*starflake schema*). Siinä hierarkiatasoja sisältävät tiedot on hajotettu omiin tauluihinsa. Tämä malli saattaa näyttää järkevältä, mutta Kimballin (1996, s.95) mukaan se on turhaa, sillä dimensiotaulut ovat huomattavan paljon pienempiä kuin faktataulut eikä tilan säästäminen kannata tällaisissa tapauksissa. Tämän lisäksi tähtimallin hajottaminen vaarantaa hakuprosessin tehokkuuden.



Kuvio 1. Star shema -tietorakennemalli (Inmon, 1997a, s.218)

Tähtihiutalemalli sisältää osia tähtimallista ja lumihiutalemallista sekä joitakin uusia ideoita (Anahory, 1997, s.81-83). Tätä mallia perustellaan sillä, että todellisessa elämässä on usein hyvin vaikea sovittaa tietorakenteita tähti- ja lumihiutalemalleihin. On hyvin tavallista, että joukko kohteita tai relaatioita ulottuu yhden tai useamman dimension yli. Esimerkiksi laajemmassa jälleenmyyntijärjestelmässä on tavallista, että paikallisilla kaupoilla on omat hinnat tuotteilleen. Tällöin hinnan hakeminen tapahtuu tuotetaulun sijasta tuote- ja kauppataulujen leikkauksesta. Tähtihiutalemalli hyväksyy sen, että todellisessa elämässä on helpompaa ja tehokkaampaa sallia jonkinasteinen ristiveto dimensioiden välille.

Dimensiotaulut eivät sisällä varsinaista haettavaa tietoa, vaan näkökulmia faktatauluun pääsemiseksi. Esimerkkinä faktataulun tiedoista voidaan mainita tuotteen

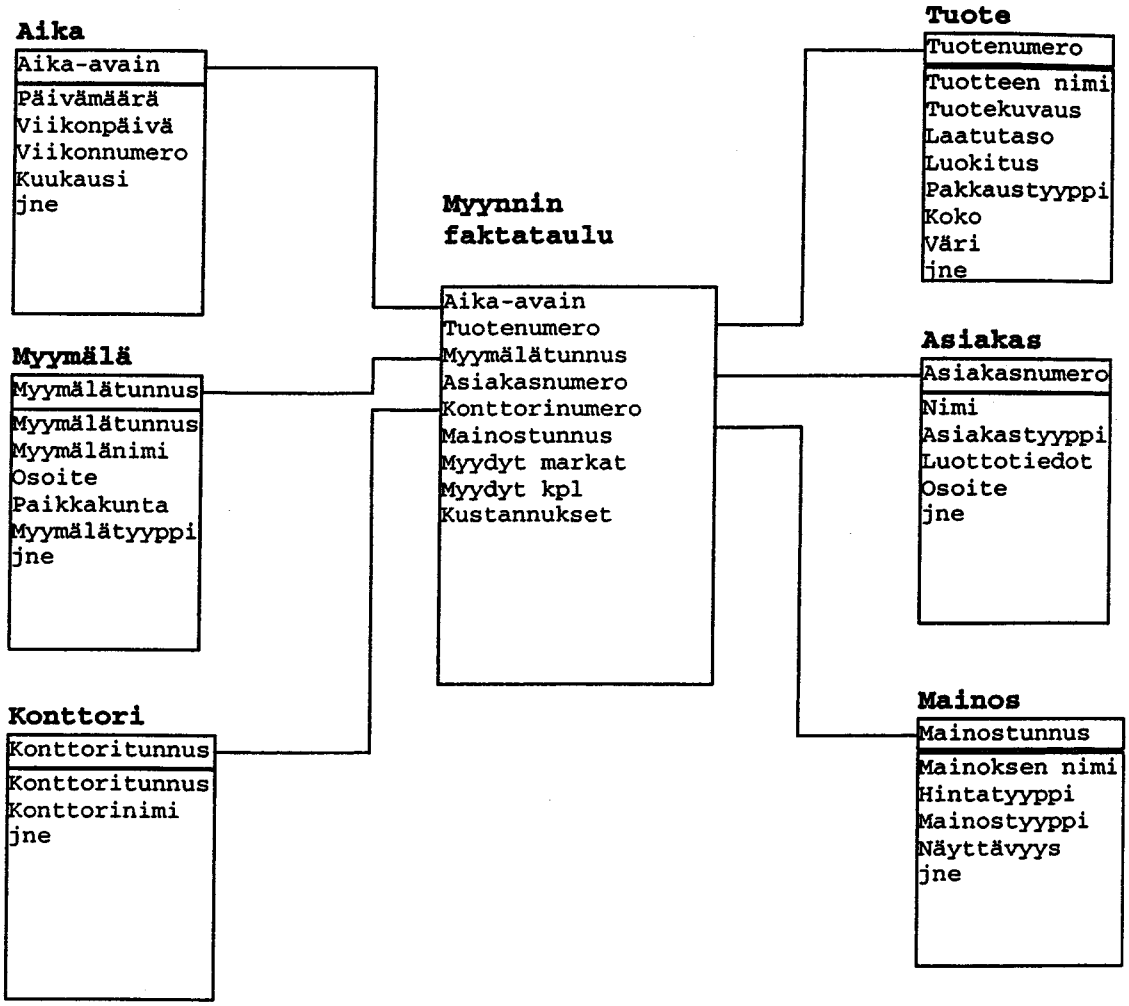
myyntimäärä, tuotteen myynnin markkamäärä ja myyntikate. Dimensiotauluja on niin monta kuin on näkökulmia haettavaan tietoon. Esimerkkejä dimensiotauluista on asiakkaat, toimittajat, tuotteet ja aika. Tietovarastossa aika on aina yksi ulottuvuus ja sillä kytketään faktataulussa olevat tiedot tiettyyn ajanhetkeen. Dimensiotaulut eroavat faktataulusta siinä, että sen tiedot muutetaan (Anahory, 1997, s.43).

Tietovaraston tiedon haussa periaatteena on valita hakukriteerit dimensiotaulujen avaimia käyttäen ja edetä niillä kriteereillä faktatauluun. Mitä useammalla dimensiotiedoilla faktataulun tietoja haetaan, sitä paremmat mahdollisuudet on rajata kyselyä niin, että luettava tietuemäärä on mahdollisimman pieni.

Fakta- ja dimensiotaulujen lisäksi tietovarastoon kuuluu tietojen haun nopeuttamiseksi valmiiksi summattua tietoa. Useimmat käyttäjät eivät tarvitse yksityiskohtaista tietoa, vaan tietyllä tasolla summatut tiedot riittävät. Kun tietovarastossa on etukäteen summattuja tietoja, niiden haku on erittäin nopeaa, koska summausta ei tarvitse tehdä perinteisellä tavalla lukemalla kaikkia yksittäisiä tietueita samalla summaten niitä.

Kellyn (1996, s.95) mukaan tietovaraston tiedon käyttö vaihtelee organisaation eri tasojen välillä. Työntekijätaso tarvitsee pääasiassa yksityiskohtaista ja yksittäisten prosessien kontrollointiin tarvittavaa tietoa. Kun organisaatiopyramidia noustaan ylöspäin keskijohtoon ja ylempään johtoon, tietoa tarvitsevien henkilöiden lukumäärä pienenee. Samalla tietotarve keskittyy enemmän summattuun ja ulkopuoliseen tietoon, sekä toiminnan suunnittelussa tarvittavaan tietoon.

Seuraavassa (kuvio 2) on esimerkki fakta- ja dimensiotaulujen tietosisällöstä, kun kohteena on myyntisovellus. Faktataulu sisältää avaimet kaikkiin dimensiotauluihin ja itse talletettavan tiedon. Dimensiot, joista käsin itse tietoon päästään käsiksi, sisältävät tietoa ajasta, toimipisteestä, myymälästä, asiakkaasta, tuotteesta ja mainostuksesta. Kun halutaan hakea tietoa tietyn tuotteen myynnistä määrittelemättä mitään muita kriteereitä, voidaan tietojen hakuun lähteä tuote -



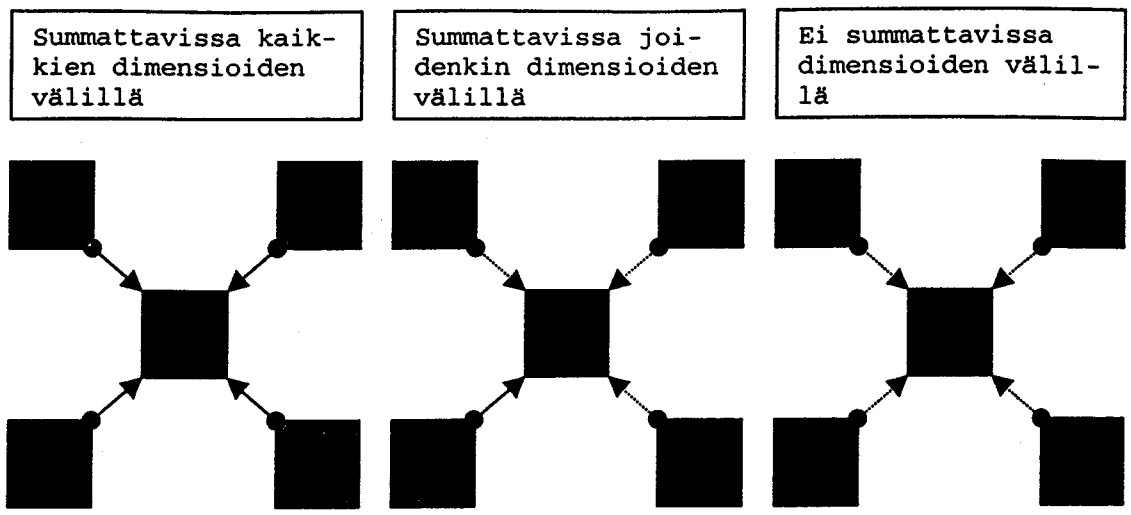
Kuvio 2. Esimerkki fakta- ja dimensiotaulujen sisältämästä tiedosta

dimensiotaulusta käsin antamalla tuotenumero. Tuotenumeron perusteella haetaan myynnin -faktataulusta kaikki halutun tuotteen myyntiin liittyvät tietueet. Tietueita tulee tällöin suuri määrä, sillä mitään muita rajaavia tekijöitä ei tässä ole annettu. Kun hakua rajataan muidenkin dimensiotaulujen arvolla, haettava tietuemäärä putoaa oleellisesti. Kun halutaan hakea kyseisen tuotteen myynti asiakkaan X osalta niin, että mainoskampanjan Y vaikutus myymälän Z myyntiin heinäkuun ensimmäisellä viikolla vuonna 2000 on selvitettävä, niin haku sisältää jo rajaavia tekijöitä useimpiin dimensioihin. Tällöin tuote-, asiakas-, mainos- myymälä- ja aikadimensioiden avulla tehtävä rajaus ja haku faktatauluun antaa täsmällisen tuloksen mainoskampanjan aikaisesta myyntimäärästä halutussa myymälässä.

2.2.2 Faktataulut

Faktataulu sisältää tietovarastossa itse tiedon. Faktataulun tiedot ovat yleensä summattavissa olevia numeerisia arvoja. Kimballin (1996, s.12) mukaan parhaimmat ja käyttökelpoisimmat faktataulun tiedot ovat summattavissa olevia ja jatkuvia tietoja, sillä tietokannanhallintajärjestelmät osaavat käsitellä niitä parhaiten, ja ne ovat aina summattavissa helposti muutamiksi vastausriveiksi kyselyissä. Faktataulu sisältää myös viiteavaimet dimensiotauluihin, joita hyväksi käyttäen tieto saadaan esille. Faktataulun tieto voi olla summattavissa (*additive*), osittain summattavissa (*semiadditive*) tai ei-summattavissa (*non-additive*) (Kimball, 1996, s. 13). Kun kenttä on summattavissa, niin kenttä voidaan summata kaikkien dimensioiden suhteen. Tässä tapauksessa faktataulun kenttään johtaa polku kaikkien dimensiotaulujen kautta. Osittain summattavien kenttien faktataulun tieto on laskettavissa vain joidenkin dimensioiden kesken. Tällöin tietoon ei löydy hakupolkua kaikkien dimensiotaulujen kautta. Ei-summattavissa olevien kenttien tiedot ei ole suoraan laskettavissa minkään yksittäisen dimension suhteen (esimerkiksi myyntikate). Tällöin niistä yksittäisinä tietoina ei voi saada ulos muuta kuin lukumäärätietoa.

Faktataulun summattavuusvaihtoehdot eri dimensioiden välillä on kuvattu kuviossa 3. On olemassa paljon liiketoimintaprosesseja, jotka on kannattavaa esittää tietovarastossa, vaikka ne eivät sisältäisikään varsinaista faktataulussa talletettavaa tietoa. Tällaisia tietoja kutsutaan *factless fact tables*. Tällaiset tiedot eivät sisällä faktataulussa mitään mitattavaa attribuuttia, mutta sinne tallennetaan kuitenkin jotain tietoon liittyvää. Kimballin (1996, s. 143) mukaan näitä tapauksia on kahdenlaisia. Tapahtumapohjaiset taulut (*event tracking tables*) ovat tapauksia, joissa kaikki haettava tieto löytyy dimensiotauluista. Tällöin faktatauluun tallennetaan vain teennäinen tiedon olemassaoloa kuvaava avain (*artifact exsistence key*) arvoilla 0 ja 1. Raportointi- tai selostamistaulut kuvaavat tapahtumaa, joka ei ole



Kuvio 3. Faktataulun sisältämien tietojen summattavuusvaihtoehdot tietovaraston eri dimensioiden välillä

tapahtunut. Tässä tapauksessa tietovarastosta voidaan hakea tietoa tapahtumista, joita ei ole tapahtunut. Esimerkiksi kaupan tietovarastosta voitaisiin hakea tietoa niistä alehintaisista tuotteista, joita ei ole ostettu.

Faktataulun sisältämä tiedon määrä riippuu tietovaraston suunnitteluvaiheessa päätetystä rakeisuusasteesta (*granularity*). Tämä määrää myös sen ajallisen tarkkuustason, millä tietoja on mahdollista hakea tietovarastosta. Rakeisuus kuvaa sitä aikaulottuvuuden tarkkuutta, millä tiedot tallennetaan faktatauluun. Aika tallennetaan tietovarastosovelluksissa aina ainakin faktatauluun. Aika voidaan tallentaa faktataulun lisäksi myös omaksi dimensioksi, jolloin tiedon haun ei tarvitse perustua ainoastaan kalenteripäiviin. Edellisten lisäksi myös summataulut voivat sisältää ajallisen ulottuvuuden.

Inmon (1996, s.51) esittää kaksitasoisen rakeisuustason. Tässä mallissa aktiivisesti käytettävä tieto on tietovarastossa, ja tähän kohdistuu n.95% prosessoinnista. Vanhentunut tietomassa on toisella tietovälineellä, johon ei välttämättä ole tosiaikaista yhteyttä, mutta josta se on helposti haettavissa samassa muodossa kuin

se on itse tietovarastossakin. Se aikaväli, jona tieto täytyy olla nopeasti saatavilla, vaihtelee eri sovelluksissa. Esimerkiksi palkanlaskentajärjestelmissä sen täytyy olla kuluva ja edellinen palkanlaskentakausi, sillä kuluvaa tilikautta tallennetaan parhaillaan ja edellisen tilikauden juuri maksetut palkat ovat vielä työntekijöiden tarkastelun kohteena.

Anahoryn (1997, s.72) mukaan aika voidaan tallentaa faktatauluun kolmella erilaisella tekniikalla. Yksinkertaisin ja helpoin tapa käyttää aikaa faktataulun tietona on tallentaa aika päivämääräkenttänä kuhunkin faktataulun tietueeseen. Toinen vaihtoehto tulee kyseeseen silloin, kun faktataulu partitioidaan ajan suhteen. Tällöin erillisien partitioiden sisällä aikaan voidaan viitata indeksinumerolla suhteessa partition alkamishetkeen. Tiedot voidaan partitioida esimerkiksi kuukausittain, jolloin partitio sisältää ainoastaan tietyn kuukauden tiedot. Esimerkiksi kyselyssä, joka sisältää kahden kuukauden tiedot, muodostetaan kaksi erillistä relaatiota, jotka yhdistetään. Kuukauden päivän numero voidaan tallentaa tilaa säästäen pelkällä indeksinumerolla, jolloin indeksin arvolla nolla viitataan kyseisen kuukauden ensimmäiseen päivään jne. Ajan viittaukseen perustuvalla tallennustavalla on Anahoryn mukaan (1997, s.73) ongelmana tiedon hakemisessa vaadittava indeksin ja päivän konversio. Kolmas tapa on tallentaa faktatauluun aikaväli. Aikavälin käyttäminen on kätevä tapa vähentää tietueiden määrää, sillä uusi tietue kirjoitetaan vasta, kun joku tieto muuttuu. Tällöin alkuperäinen tietue saa loppupäivämäärän ja uusi kirjoitetaan tietovarastoon. Tällä tekniikalla voidaan Anahoryn mukaan (1997, s.74) vähentää tietueiden määrää oleellisesti: jos 15% koko faktataulun tietueista muuttuu, aikavälin käyttäminen pienentää faktataulun kokoa 18% verrattuna siihen, että tiedot tallennettaisiin erillisinä. Aikavälin käyttämisellä on kuitenkin huonot puolensa. Jos tietovaraston tiedon kyselyohjelmisto ei hallitse karteesista tuloa, voi aikavälin liittäminen aikadimensioon aiheuttaa suurta konekuormaa ja väliaikaisen muistin käyttötarvetta. Tällöin hakutehokkuus on huono.

2.2.3 Dimensiotaulut

Dimensiot ovat viitetietoja, joita hyväksikäyttäen faktataulussa oleva tieto yksilöidään. Dimensiotiedot tallennetaan dimensiotauluihin, joita on aina useita riippuen niistä näkökulmista, joilla itse faktataulun tietoa halutaan lähestyä. Dimensiot muistuttavat sql-lauseen hakua rajaavia where -ehtoja. Dimensiot eivät käyttäydy latausajoissa samalla tavalla kuten faktataulut. Kun faktatauluun tallennetaan uusia tietoja kytkettynä tiettyyn aikaulottuvuuteen, dimensiotietojen muutosten on pystyttävä kuvaamaan vastaavat muutokset. Dimensiotiedot muuttuvat liiketoiminnan mukaan. Esimerkiksi osastojen numerointi ja tuotteiden tuotenumerot muuttuvat, jolloin tietovaraston on kyettävä kuvaamaan eri ajanhetkinä olleet tilanteet. Dimensiotaulujen muutosten hallintaan on useita menetelmiä riippuen haluttavasta lopputuloksesta.

Yksi vaihtoehto muuttuvien dimensiotietojen hallintaan on sijoittaa ne faktatauluun tai antaa näille dimensioille aikaulottuvuus. Kimballin (1996, s. 100) mukaan tämä ei ole järkevää, sillä dimensiotiedot ovat suhteellisen muuttumattomia. Tällaisia dimensiotietoja kutsutaan *hitaasti muuttuviksi dimensioiksi* (*slowly changing dimensions*). Näiden käsittelyyn Kimball esittelee kolme erilaista vaihtoehtoa. Ensimmäisessä vaihtoehdossa vanhoja dimensiotietoja ei säilytetä vaan ne korvataan uusilla. Tässä tapauksessa historiatiedot menetetään. Toisessa vaihtoehdossa luodaan uusi dimensiotietue sen ajanhetken mukaan, jolloin uudet attribuutin arvot astuvat voimaan. Tätä varten täytyy luoda uusi dimensioavain, joka poikkeaa alkuperäisestä versionumeroinnilla. Kolmas vaihtoehto, jota käytetään harvemmin, on luoda uusi kenttä ylimääräisen tietueen sijaan. Kenttä kuvaa nykyistä tilaa. Aikaisemmat tilat säilytetään ja tallennetaan omaan kenttäänsä. Tästä esimerkkinä on siviilisääty: naimaton -tieto ja päivämäärä säilytetään nykytilaa esittävässä kentässä. Kun siviilisääty muuttuu tilaan "naimisissa", säilytetään aikaisempi tieto toiseen kenttään päivämäärällä varustettuna ja nykyinen tila nykytilaa kuvaavaan kenttään.

Myös Anahory (1997, s.79) suosittelee Kimballin toista vaihtoehtoa. Hänen mukaansa uuden aikavälin lisääminen tähtimallin mukaiseen tauluun on tehokasta. Anahoryn mukaan monipuolisempi aikaulottuvuus kannattaa ottaa käyttöön silloin, kun halutaan huomioida kiinteiden kalenteripäivien sijaan tapahtumia joustavasti. Esimerkiksi kaupan kesäalennusmyynti voi tapahtua eri vuosina eri aikaan, jolloin kiinteät kalenteripäivät eivät sovellu. Tällöin täytyy muodostaa oma dimensio, aikadimensio. Tälle aikadimensiolle annetaan tietoina erilaisia tapahtumia, jotka sidotaan tässä dimensiotaulussa haluttuun aikaväliin. Tietojen haku tapahtuu tässä tapauksessa yhdistämällä aikadimension haluttu tapahtuma haluttuun kyselyyn.

Tietovaraston kaikkein oleellisin dimensio on aika. Faktatauluun tallennettu aika määrää tiedon tallennustarkkuuden eli rakeisuuden (*granularity*). Aikaulottuvuutta voidaan käyttää monipuolisesti hyväksi aikadimensiotaulun avulla. Päivämäärälle ja jopa kellonajalle voidaan antaa aikadimensiotaulussa erilaisia merkityksiä, joiden kautta faktataulun tiedot saadaan parhaimmalla tavalla esille. Aikadimensio muistuttaa monipuolista kalenteria, jossa eri ajanhetkille on annettu merkityksiä. Anahoryn mukaan (1997, s.80) päivämäärä voidaan sitoa kuuluvaksi esim. vuoden tiettyyn viikkoon, vuoden neljännekseen, vuosipuoliskoon tai tilikauteen. Päivämäärä saattaa kuulua myös juhlapyhään tai alennusmyyntikauteen. Esimerkiksi eri vuosina tapahtuneita kesäalennusmyyntejä voidaan vertailla aikadimension kautta helposti vaikka niiden kalenteripäivät eivät olisikaan eri vuosina samat.

2.2.4 Summataulut

Tietovaraston suunnittelijan tehokkain tapa tehostaa tietovaraston käyttöä on käyttää tietovarastoon etukäteen tallennettuja summatietoja (Kimball, 1996, s. 190). Summatiedot voidaan tallentaa tietovarastoon joko erillisinä tietueina fak-

tatauluun tai tekemällä erilliset summataulut (Data Warehousing fundamentals Volume 1-3, 1996, s. 4-34, Kimball, 1996, s. 191). Sekä summatietojen tallennus faktatauluun muiden tietojen sekaan että erilliset summataulut vaativat luonnollisesti viittauksen yhdestä tai useammasta dimensiotaulun tietueesta. Erillisiä summatauluja sanotaan joskus johdetuiksi faktatauluiksi, koska taulun yksi tietue voidaan johtaa alkuperäisen faktataulun tietuejoukosta. Jos summatietue tallennetaan faktatauluun, täytyy tämän summatietueen summaustaso ilmoittaa erillisellä taso -kentällä (*level field*). Taso -kenttä tallennetaan dimensiotaulun siihen tietueeseen, jolla viitataan tämän faktataulun summatietueeseen. Tässä tapauksessa kaikissa tietovaraston summatietokyselyissä on huomattava määritellä taso -tieto, sillä muuten vaarana on vastauksena saatujen tietojen moninkertaistuminen.

Summatietojen tallennusta erillisiin summatauluihin suositellaan summatietueiden sijasta, sillä tällä vältetään virheelliset tietovarastokyselyt ja voidaan hallinnoida summatauluja erillisinä tauluina helpommin: taulun luonti, poisto, lataus ja indeksointi voidaan tehdä erillään faktataulusta riippumattomana. Erilliset summataulut on helpompi kuvata metadataan, ja haku niistä on nopeampaa, sillä tietuemäärä on rajoitettu. Lisäksi summataulun tietosisältö voidaan tehdä pelkästään summattua tietosisältöä palvelevaksi. Hovin (1997, s. 85) mukaan tavanomaisin tapa on tehdä omat summataulut, jotka pysyvät normaalisti pieninä ja nopeina hakea tietoa.

Summatauluja käytetään nopeuttamaan tiedon hakua. Anahoryn (1997, s.47-48) mukaan summataulujen rakentaminen kasvattaa operatiivisia kustannuksia, sillä summatiedot on päivitettävä jokaisen latauksen yhteydessä. Summatauluja ei tarvitse Anahoryn mukaan periaatteessa varmistaa, sillä ne ovat rakennettavissa aina uudestaan. Näin ei käytännössä kuitenkaan aina ole. Anahoryn mukaan summatauluja tulee muodostaa vain tarpeen mukaan, sillä ne hidastavat latausrutiinia. Nyrkkisääntönä voidaan sanoa, että kaksisataa summataulua on ehdoton yläraja; sen jälkeen järjestelmä muuttuu kontrolloimattomaksi ja latauksen aikaikkuna saattaa jäädä liian lyhyeksi.

Hovin (1997, s. 84) mukaan summaamisella saavutetaan seuraavat edut. Tietovaraston kyselyssä tietomassa pienenee huomattavasti, jolloin CPU ja I/O -kuorma pienenevät ja vastausajat paranevat huomattavasti. Summataulut luodaan usein yhdistelemällä tietoja useasta taulusta liitoksilla. Kun useiden eri taulujen tietoja on näin jo valmiiksi yhdistelty, sitä ei tarvitse tehdä enää kyselyssä, jolloin haku nopeutuu. Yleisesti käytettävät perustiedoista johdetut tiedot talletetaan summausprosessissa valmiiksi samoilla periaatteilla laskettuina tietovarastoon. Tämä yhtenäistää samojen käsitteiden käyttöä ja nopeuttaa johdetun tiedon käsittelyä. Hovin mukaan monimutkaisten summausalgoritmien toiminta varmentuu, kun ne tehdään yhdessä paikassa keskitetysti. Lisäksi näin päästään käyttämään helppokäyttöisiä ja voimakkaita raporttigeneraattoreita ja kyselyvälineitä.

Tiedon summaaminen ei ole ongelmatonta. Summaaminen vie osansa latausprosessointiajasta, taulut vievät tilaa ja lukuisat eri summataulut lisäävät järjestelmän monimutkaisuutta entisestään. Summauskriteerit ja detaljitason tietojen säilytysaika tulee miettiä tarkkaan, sillä summauksen jälkeen mahdollisesti poistettu detaljitason tietomassa ei ole enää saatavissa. Lisäksi tietojen eheys voi vaarantua tietojen summauksen yhteydessä. Inmonin (1996, s.51) mukaan tietovarastossa tulee hyvin nopeasti tarpeen periaate, jossa detaljitason tietoja säilytetään vain rajattu aika. Tietyn ajan jälkeen tiedot summataan ja detaljitason tietomassa siirretään toiselle tietovälineelle. Tällöin tiedot eivät rasita itse tietovaraston aktiivitietokantaa, mutta ovat kuitenkin vielä pienellä vaivalla saatavissa jopa samassa muodossa kuin aikaisemmin tietovarastossakin.

Detaljitason tietomassan summaaminen tehdään normaalisti silloin, kun tiedot ladataan operatiivisesta järjestelmästä tietovarastoon. Summaaminen kuluttaa koneresursseja ja hidastaa lataamista. Summausprosessia on tämän vuoksi yritetty kehittää nopeammaksi ja helpommaksi. Uusimmat tietokannan hallintajärjestelmät pystyvät automaattisesti muodostamaan summaustasot. Tästä esimerkkinä on Oraclen 8i. Myös erilaisilla summaustekniikoilla voidaan vaikuttaa summauksen tehokkuuteen. Kaikkein yksinkertaisin summaustapa on yksinker-

tainen kumulatiivinen summaus (Inmon, 1996, s. 61-66). Siinä operatiivisesta järjestelmästä luettu tieto kumuloidaan tietovarastoon päivittäin tietue kerrallaan (*simple cumulative*). Tietueet kumuloidaan halutulla kriteerillä päivittäin, kuukausittain ja vuosittain tarpeen mukaan. Tässä tapauksessa kaikki kumulointi tapahtuu tietue kerrallaan. Hieman kehittyneempi tapa on käyttää kiertävää kumulatiivista summausta (*rolling summary*). Tässä tapauksessa tiedot ladataan tietovarastoon kuten edellisessä tapauksessakin, mutta päivätason summatiedoista kumuloidaan viikkotason summatiedot, viikkotason summatiedoista kumuloidaan kuukausitason summatiedot, kuukausitason summatiedoista saadaan edelleen vuositason tiedot ja niin edespäin. Erona edelliseen tapaan on se, että jo kumuloituja tietoja voidaan käyttää syöttötietona seuraavan tason kumulointiin. Tämä nopeuttaa oleellisesti kumulointia, sillä silloin ylemmän tason kumuloinnissa vältetään detaljitason tiedon käsittely. Toisen tyyppinen ratkaisu on yksinkertainen suora tiedosto (*simple direct file*). Tässä tapauksessa operatiivisen järjestelmän tietueet ladataan tietovarastoympäristöön sellaisenaan summaamattomana. Tiedoista voidaan tämän jälkeen muodostaa ns. tietojen jatkuva tiedosto (*continuous file of data*) yhdistämällä jatkuvasti esimerkiksi kuluvan kuukauden tiedot alkuvuoden jatkuvaan tiedostoon. Avaintietona joudutaan lähes poikkeuksetta käyttämään koottuja avaimia (*compounded key*), sillä aika on lähes aina avaimen yksi osa. Jos tietovaraston taulut on partitioitu, partitiointiavain on usein avaimen osana.

2.3 Tiedon moniulotteinen käsittely

OLAP eli *online analytical processing* tarkoittaa moniulotteisen tiedon analysointia. Inmonin (1997a, s.183) mukaan on joukko ohjelmistoteknologioita, jotka antavat mahdollisuuden analyytikoille, johtajille ja muille toimeenpaneville henkilöille saada nopea, yhdenmukainen ja vuorovaikutteinen katsaus laajaan käyttäjien ymmärtämään ja liiketoimintaa vastaavaan tietomassaan. Tosiaikaisen tiedon ja-

lostus on luonnollinen tietovaraston laajennus. Lyhyesti sanottuna tosiaikaisen tiedon jalostus on joukko toiminnallisuuksia, jotka yrittävät helpottaa moniulotteista analysointia. Moniulotteinen analysointi on kyky käsitellä tietoa, joka on yhdistetty erilaisiin dimensioiden luokkiin (Inmon, 1997a, s.183).

Tekniikoita, joilla moniulotteinen tieto saatetaan käyttäjän näkemäksi on kehittynyt useita. ROLAP, *relational online analytical processing* ei tarvitse käsittelyssään kuutioita vaan tieto muokataan moniulotteisen näköiseksi. MOLAP, *multidimensional online analytical processing* vaatii, että kuutiot ovat rakennettu. Tämä on tehokkain moniulotteisen tiedon hakutapa. HOLAP, *hybrid online analytical processing* katsotaan sisältävän sekä MOLAP että ROLAP –ominaisuudet. Näitä menetelmiä, joilla normaali relaatiotietokanta saadaan käyttökelpoisemmaksi moniulotteiseen tiedon hakuun sanotaan teho-relaatiotietokantajärjestelmäksi (*super-relational database system*) (Jarke, 2000, s.7).

Hovin mukaan (1997, s.60) tosiaikaisen tiedon jalostustuotteiden tärkeimmät toiminnot ovat porautuminen (*drill down*), karkeistaminen (*roll up*), hierarkioiden käsittely, parhaat arvot -kyselyt (*top ten*), kyselyt eri luokittelutekijöillä, summien katselu eri hierarkiatasoilla, ristiintaulukointi ja erilaiset graafiset tuotteet. Nykyisten monipuolisten kyselytyökalujen perusominaisuuksiin moniulotteisessa haussa kuuluvat viipalointi (*slicing*), kuutiointi (*dicing*) ja porautuminen (*drilling*) (Data Warehousing fundamentals Volume 1-3, 1996, s. 13-31). Viipaloinnissa otetaan tietomassasta haluttu otos attribuuttien leikkauksena. Esimerkiksi haku voi antaa tuloksena asiakkaan tiedot. Kuutiointi sallii ottaa viipaleen eri suunnasta. Esimerkiksi haettu otos käsitellään vielä uudelleen antamalla sille lisämääreitä, ja näin saadaan entistä suppeampi ja tarkkaan määritelty hakutulos. Näitä kahta käytetäänkin normaalisti yhdessä (*slicing and dicing*). Porautuminen voi tapahtua ylhäältä alas (*drill-down*), alhaalta ylös (*drill-up*) tai vaakatasossa (*drill-accross*). Vaakatasossa tapahtuva kysely tapahtuu siirtymällä faktataulusta toiseen samassa kyselyssä.

2.3.1 Kuutiot

Kuutiot mahdollistavat moniulotteisen tiedon haun tietovarastosta. Kuutiot rakennetaan ohjelmistotoimittajan välineillä usein relaatiotauluista tietoa keräämällä ja tallettamalla ne muotoon, josta moniulotteinen haku on mahdollista. Tällöin muodostuneet kuutiot ovat varsinaisesta tietokannasta erillään olevia objekteja, joita ei normaalisti ylläpidetä vaan rakennetaan aina tarvittaessa uudestaan. Kuutiot ovat usein suuria ja tilaa vieviä. Myös niiden rakentaminen vie runsaasti aikaa ja koneresursseja. Tiedon hakeminen tietovarastosta ei kuitenkaan vaadi aina kuutioita vaan haussa voidaan käyttää myös monien ohjelmistotoimittajien valmiita raportointityövälineitä.

Jotta tietovarastoon saataisiin moniulotteisen analysoinnin mahdollisuudet (Inmon, 1997a, s.213), lähestymistapa voi olla yritys hallita tähtimalli osastotasolla, käyttää tosiaikaisen tiedon jalostusvälineitä (*OLAP*) normalisoidun tiedon hakemiseen relaatiokannasta tai käyttää osastotasoista moniulotteista tietokantaa. Tällöin tietovaraston arkkitehtuuri voi olla kaksi- tai kolmitasoista (Inmon, 1997a, s.214-216). Kaksitasoisella arkkitehtuurilla tarkoitetaan yleensä relaatiotietokantaan perustuvaa tosiaikaisen tiedon jalostusta (*ROLAP*), jossa moniulotteinen analyysi tehdään työaseman päässä ja relaatiotietokanta on tietokantapalvelimella. Jotkut moniulotteisen analyysin välineet pystyvät hakemaan tietoja suoraan normalisoidusta relaatiokannasta, jotkut taas vaativat toimiakseen tähtimallin. Kolmitasoinen arkkitehtuuri vaatii työaseman ja tietokantapalvelimen väliin vielä lisäksi osastokohtaisen palvelimen, jonka tehtävä on ylläpitää ja tukea tietovaraston osastokohtaista esimerkiksi maantieteellistä jakoa.

E. F. Codd on määritellyt vuonna 1993 kaksitoista tosiaikaisen tiedon jalostussääntöä (OLAP-säännöt), joilla määritellään moniulotteisen tiedon käsittelyn periaatteellisia vaatimuksia, jotka Hovi (1997, s.59) esittelee:

1. Moniulotteinen käsitteellinen näkymä. Tiedot nähdään moniulotteisena mallina; voi ryhmitellä, summailla ja porautua.

2. Läpinäkyvyys. Kaksitoista tosiaikaisen tiedon jalostusvälineen sijainnin pitää olla tuntumatonta tai näkymätöntä käyttäjälle.
3. Saatavuus. Tosiaikaisen tiedon jalostusvälineen fyysinen rakenne on piilotettu käyttäjältä ja välineen pitää hakea ja muotoilla tiedot käyttäjälle sopivaan, yhdenmukaiseen muotoon.
4. Tasainen raportoinnin suorituskyky. Raportoinnin suorituskyky ei saisi heikentyä, kun ulottuvuuksien määrä kasvaa.
5. Työasema/palvelin -arkkitehtuuri. Tosiaikaisen tiedon jalostustuotteen palvelinosan tulee olla joustaja, jotta siihen voi kytkeytyä erilaisilla työasematuotteilla.
6. Yleinen moniulotteisuus. Ulottuvuuksien pitää olla yhdenmukaisia rakenteeltaan ja käsittelyominaisuuksiltaan; funktioiden pitää olla toimivia kaikilla ulottuvuuksilla.
7. Dynaaminen harvan matriisin hallinta. Tosiaikaisen tiedon jalostusvälineen tulee hallita harvan matriisin käsittely optimaalisesti ja pystyä tallettamaan tiedot tehokkaasti.
8. Monen käyttäjän ominaisuudet. Tosiaikaisen tiedon jalostusvälineen tulee tarjota tukea monen käyttäjän yhtäaikaiselle käytölle.
9. Rajoittamattomat ulottuvuuksien väliset operaatiot. Tuotteen tulee tarjota valmiita laskentatapoja kaikille ulottuvuuksille sekä mahdollisuuden määritellä omia laskentatapoja kaikille ulottuvuuksille ilman rajoituksia.
10. Intuitiivinen käsittely. Käsittelytavat, kuten porautumisen, tulee tapahtua helposti ilman monimutkaisia operaatioita käyttöliittymässä.
11. Joustava raportointi. Käyttäjän tulee tosiaikaisen tiedon jalostusvälineen työkaluilla voida käsitellä ja analysoida tietoja kaikilla haluamillaan tavoilla, kuten muodostamalla loogisia ryhmittelyjä tai siirtelemällä rivejä toistensa viereen.
12. Rajoittamaton määrä ulottuvuuksia ja summaustasoja. Tosiaikaisen tiedon jalostuspalvelimen tulee tukea vähintään 15 ulottuvuutta. Kunkin ulottuvuuden tulee sallia rajoittamaton määrä käyttäjän määrittelemiä summaustasoja.

2.3.2 Data mining

Tiedon louhinta *(data mining)* on monimutkaista tiedon hakemista tietovarastosta. Periaatteena on analysoida tietoa annettujen kriteerien perusteella. Tiedon "louhinta" hakee lainalaisuuksia, syitä ja seurauksia erilaisiin asioihin. Tällainen tiedon haku on ainoastaan harvojen käyttäjien käytössä, sillä kyselyt saattavat kestää tunteja ja saatujen tulosten analysointi vaatii asiaa tuntevan ammattilaisen.

Rob Mattison (1997, s.96) jakaa tiedon hakemisen sekä toiminnalliseen (*operational monitoring and control*) että tutkimuskäyttöön. Toiminnallinen haku antaa käyttäjälle tietoa liiketoiminnasta: mitä tapahtuu, miksi, missä ja milloin. Tähän kuuluvat kysely- ja raportointityökalut, tosiaikaisen tiedon jalostus (OLAP), hakuagentit ja web-työkalut. Tutkimuskäytössä tarkoituksena on se, että järjestelmä opettaa jotain uutta käyttäjälle. Tähän kuuluvat Mattisonin mukaan tilastolliset analysointityökalut, tiedon hakutyökalut (*data discovery tools*) ja tiedon visuaalisen esityksen työkalut.

2.4 Metadata

Jotta tietovaraston tietoja voidaan käyttää täydellä teholla hyväksi, täytyy löytyä tietoa siitä, mitä tietovarasto sisältää. Metadata on tietoa siitä tiedosta, mitä tietovarastossa on. Metadataa kerätään koko tietovaraston eliniän ajan: rakentamisesta sen ylläpidon vaatimiin muutoksiin ja normaalien latauskertojen dokumentoimiseen. Metadata voidaan luokitella kolmeen tyyppiin (Data Warehousing fundamentals Volume 1-3, 1996, s. 4-39): tiedon keräämiseen, käsittelyyn ja tietovarastoon siirtämiseen liittyvä metadata, loppukäyttäjän metadata sekä kolmantena operatiivinen metadata. Ensin mainittu ett-prosessiin liittyvä metadata sisältää

tietoa loogisesta tietomallista ja tietokannan taulurakenteista sisältäen fakta- ja summataulujen kuvaukset. Lisäksi se sisältää lähdejärjestelmän ja tietovaraston tietojen vastaavuudet sekä ne säännöt, joilla tiedot poimitaan, yhtenäistetään ja siirretään tietovarastoon. Loppukäyttäjän metadataa katsotaan tiedon analysoijan käyttöliittymän suunnasta ja sen tarkoitus on näyttää käyttäjälle tietovaraston tietojen merkitys. Sen päätarkoitus on helpottaa ja avustaa käyttäjää tietovaraston tietojen navigoinnissa. Operationaalinen metadata sisältää tietoa latausprosessista, ylläpitotoimenpiteistä ja tiedon hausta. Myös muiden tietovarastoasiantuntijoiden kirjallisuudessa on käytetty samankaltaista ryhmittelyä (ks. Devlin, 1997, s.54).

Inmonin mukaan (1997b, s. 255) yksi metadatan käyttötarkoitus on pitää yllä tietoa tietovaraston tietoihin vuosien varrella tapahtuneista muutoksista niin, että epäyhtenäisiltä näyttävät tiedot voidaan tulkita kulloisenkin ajanhetken mukaan ymmärrettäväksi. Jotta tietovaraston metadata olisi suhteellisen vaivatonta ylläpitää ja saada käsille, on suunnittelu- ja toteutusvaiheessa hyvä käyttää casevälinettä (Inmon 1997b, s. 268, Anahory, 1997, s.134), sillä se mahdollistaa järjestelmän rakennusaikaisen suunnittelutiedon näyttämisen ja ylläpidon. Caseväline mahdollistaa myös tiedon latausvaiheen puhdistus- ja muunnossääntöjen yksiselitteisen ylläpidon ja esittämisen tiedon tarvitsijoille.

Metadatan käsittelyssä on vielä tällä hetkellä suuri ongelma eri ohjelmistotoimittajien tietojen tallentamisessa. Metadata-tiedon käsittelyyn ei ole olemassa yhteisesti sovittua standardia eikä sitä tule lähiaikoina syntymäänkään (Kimball, 1996, s. 221). Hackathorn (1999, s. 169) kuitenkin esittelee The Metadata Coalitionin työn alla olevia metadata-määrittelyjä, joista esimerkiksi MDIS, Metadata Interchange Specifications, on joukko sääntöjä metadatatiedostojen käsittelyyn erilaisten tietovarastotyökalujen välille. Tämä osoittaa, että työtä ollaan kuitenkin tekemässä.

2.5 Tekninen ratkaisu

Tietovarastosovelluksessa käytettävän laitteiston tulee olla tarpeeksi vikasietoista (*robust*), helposti saatavilla (*available*), luotettavaa (*reliable*), joustavaa (*flexible*), laajennettavissa (*extensible*), skaalautuvaa (*scalable*), tuettua (*supportable*), elpyvää (*recoverable*) ja rinnakkaiskäsittelyn sallivaa (*parallel*). Lisäksi laitteiston täytyy tukea VLM-teknologiaa (*very large memory*) ja olla avoin yleisiin standardeihin (Data Warehousing fundamentals Volume 1-3, 1996, s. 11-9). Tärkeintä laitteiston valinnassa on se, että se on helposti skaalautuvaa; laitteiston on oltava laajennettavissa alati lisääntyvän levytila- ja prosessoritarpeen tyydyttämiseksi.

Tietovarastoratkaisussa voi olla eri tasoisia arkkitehtuuritasoja (Data Warehousing fundamentals Volume 1-3, 1996, s. 11-27). Tietovarasto sijoitetaan lähes poikkeuksetta omalle palvelimelleen, jolloin kaksitasoinen (*two-tier architecture*) on ensimmäinen taso. Tässä lähdejärjestelmät sijaitsevat toisella laitteistolla ja tietovarasto toisella. Kolmitasoinen arkkitehtuurirakenne (*three-tier architecture*) sisältää väliohjelmisto-ratkaisun (*middleware*) edellä mainittujen väliseen tiedon hakuun ja muunnoksiin. Tällöin kolmas taso voidaan jättää tosiajan käyttöön, raporttien käsittelyyn ja analysointiin sekä graafisen tiedon esitykseen. Nelitasoinen (*four-tier architecture*) arkkitehtuurirakenne on muuten samanlainen kuin kolmitasoinenkin, mutta lisänä siinä on Web-pohjainen taso.

Inmonin (1996, s.45) mukaan tietovaraston suunnittelussa tärkein yksittäinen asia on rakeisuus ja toiseksi tärkein on tietokantataulujen partitiointi. Nämä nousevat muiden yli siksi, että tietovaraston käytön suurin ongelma on suuren tietomäärän hallinta ja Inmonin mukaan näillä kahdella suunnittelutekijällä on asiaan merkittävä vaikutus. Rakeisuus on Inmonin mukaan tärkein tietovaraston suunnittelutekijä, joka on operatiivisissa järjestelmissä yleensä aina ollut valmiiksi annettu tieto, mutta tietovarastossa näin ei ole. Mitä enemmän tietovarastossa on tietoa, sitä alhaisempi rakeisuusaste on. Rakeisuusasteen valinta määrää täysin tietova-

rastossa säilytettävien tietojen määrän. Se määrää myös alhaisimman tason, joka on tietotasolla haettavissa tietovarastosta.

Tietovaraston suunnittelun toinen kulmakivi on Inmonin (1996, s.55) mukaan tietokantataulujen partitiointi. Kyse ei ole siitä partitioidaanko tietokannan tiedot vaan siitä, miten ne partitioidaan.

2.5.1 Rinnakkaiskäsittely

Jos laitteistossa on useita prosessoreita, rinnakkaiskäsittely (*parallel processing*) nopeuttaa sekä latausprosessia että tietokantahakuja. Rinnakkaiskäsittely on mahdollista myös indeksien, lajittelun, varmuuskopioinnin, palautusten ja taulujen luonnin yhteydessä eli yleensäkin tietokannan hallintalauseissa (Data Warehousing fundamentals Volume 1-3, 1996, s. 11-35...39). Rinnakkaisuutta voidaan käyttää hyväksi tietokannan hoidossa nopeuttavana tekijänä, vaikka koneessa ei olisi kuin yksi prosessori. Rinnakkaiskäsittely monimutkaistaa tietokannan hallintaa ja pullonkaulojen etsintää, sillä tietokannanhallintajärjestelmä jakaa tai halutessaan jättää jakamatta pyydetyn sql-lauseen useaksi rinnakkain ajettavaksi kyselyiksi. Kun prosessit ajetaan rinnakkain, niissä ei voida käyttää hyväksi tietokantataulun indeksejä, vaan haut tehdään rowid -perusteisesti jakamalla hakualue between -määreellä rajattuihin arvoväleihin. Toisaalta tietokannanhallintajärjestelmä voi järjestelmäresurssien puutteen vuoksi jättää tekemättä rinnakkaiskäsittelyn vaikka sql-lause yritettäisiinkin pakottaa siihen vinkeillä (*hint*). Rinnakkaiskäsittely on kannatettava vaihtoehto silloin, kun koneessa löytyy prosessoritehoa. Muuten ominaisuuden käyttöönotto tukkii jo muutenkin kovilla olevan koneen kokonaan aiheuttamalla osatehtävien jonotusta hitaampien rinnakkain ajettavien tehtävien suorituksen aikana, jotta osatehtävien antama välitulos saataisiin seuraavien osatehtävien syöttötiedoksi.

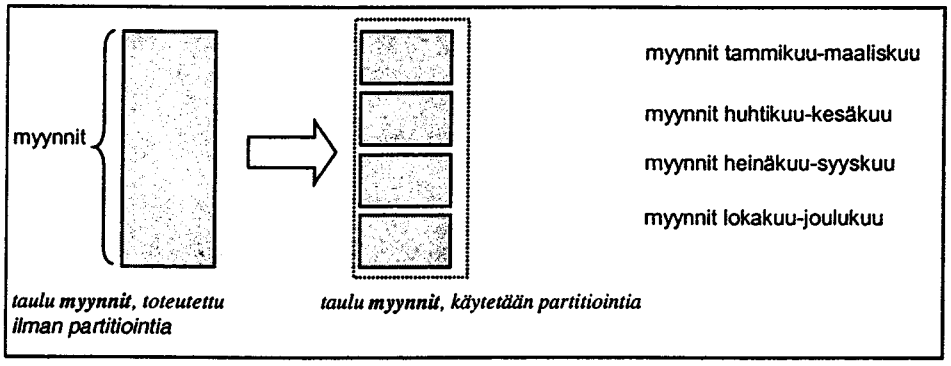
2.5.2 Ositus

Ositus eli partitiointi on yksi tietovarastoinnin oleellisimmista nopeuttavista tekijöistä. Partitiointi pitää tehdä taulujen luonnin yhteydessä ja sillä on tarkoitus jakaa suuret taulut paremmin hallittaviin osiin. Partitiointitekijän valinta on tässä oleellinen asia (Data Warehousing fundamentals Volume 1-3, 1996, s. 11-41). Partitiointi voi olla vaakaositusta (*horizontal partitioning*) tai pysytyositusta (*vertical partitioning*). Vaakaositus tapahtuu useimmin ajan suhteen. Myös organisaatio, myyntialue tai maantieteellinen sijainti voivat olla valintakriteereinä. Pystyositus jakaa taulun sen sarakkeiden mukaan eri partitioihin. Sellaiset taulun kentät, jotka ovat stabiileja ja vastaavasti toiset, jotka muuttuvat jatkuvasti, kannattaa sijoittaa omiin partitioihinsa.

Osituksella tarkoitetaan erittäin suuren taulun tai indeksin jakamista osiin eli partitioihin tehokkuuden ja käytettävyyden parantamiseksi. Pienemmissä osissa onnistuneella jakamisella kukin partitio on helpommin hallittavissa. Siihen voidaan kohdistaa kyselyitä ja päivityksiä, jolloin käsiteltävä tietuemäärä on pienempi. Se voidaan tyhjentää nopeasti, siirtää paikasta toiseen, ottaa pelkästään siitä varmuuskopio ja palauttaa se yksinään. Varsinkin sellaisissa tauluissa, joihin uutta tietoa tulee paljon ja vanhaa poistuu, on ositus erityisen tärkeä. Normaalisti tietojen kirjoitus tietokantaan on nopeaa, mutta poisto hidasta. Hitaus johtuu siitä, että poistettavat tietueet on kirjoitettava kokonaisuudessaan rollback -alueelle, jotta samanaikaiset kyselyt löytäisivät tietueen siihen saakka, kunnes poisto varmistetaan commit -komennolla. Tietokantaan kirjoittamisen yhteydessä rollback -alueelle tarvitsee kirjoittaa vain rowid -tieto. Tällöin poistetut tietueet ainoastaan merkataan poistetuiksi ja taulun viimeistä tietuetta osoittava merkki (*high water mark*) jää kuitenkin osoittamaan viimeisintä tietuetta. Kaikki kyselyt käyttävät tätä high-water-mark -merkkiä hauissaan, mikä aiheuttaa käsittelyn jyrkän hidastumisen. Vaikka esimerkiksi 1 miljoonan tietueen poisto voitaisiin tehdä yhdellä sql-komennolla, käytännössä se ei ole mahdollista vaan komento on jaetta-

va pienempiin ja hallittavampiin osiin. Rollback -alue, jolle poistettavat tietueet tallennetaan ennen varsinaista poistamista, ei useinkaan riitä noin suuren tietuemäärän poistamiselle. Käytännössä ongelmana on myös se, ettei poistettavien tietueiden lukumäärä ole edes tiedossa tai se muuttuu ajan kuluessa, jolloin rollback -alue saattaa jäädä etukäteissuunnittelusta huolimatta liian pieneksi.

Oikeanlaisella osituksella tämä voidaan estää, sillä tiedot voidaan poistaa taulusta partitiokohtaisesti TRUNCATE -komennolla, joka pudottaa high-water-mark -merkin alkuun. Lisäksi tämän tyyppinen tietojen poisto on erittäin nopea, koska silloin tiedoston loppumerkki ainoastaan siirretään alkuun eikä tietoja käsitellä tietue kerrallaan. Kuviossa 4 on kuvattu osituksen periaate osittamalla myyntitaulu kuukausien mukaan neljään eri partitioon.



Kuvio 4. Myynnit -taulun jako neljään partitioon

Ositus voidaan tehdä kolmella eri tavalla: arvoalue- (*range partition*), hajautusosituksella (*hash partition*) tai yhdistelmäosituksella (*composite partition*).

2.5.2.1 Arvoalueositus

Taulun tai indeksin ositus tehdään sarakearvojen perusteella, jolloin partitiointiavain voi muodostua yhdestä tai useammasta sarakkeesta. Joissain tapauksissa (esimerkiksi Oracle 8.0) partitioavaimeen kuuluvia sarakkeita ei voi päivittää,

mikäli päivitys aiheuttaisi rivin siirtymisen toiseen partitioon. Ongelma voidaan kiertää poistamalla vanha rivi ja luomalla uusi. Aikaan perustuvat sarakkeet (esimerkiksi myyntipäivämäärä) ovat usein selkeitä partitiointiavaimia eli taulu voidaan helposti jakaa loogisiin kokonaisuuksiin ajan perusteella, vaikkapa kolmen kuukauden jaksoihin, joista kukin muodostaa yhden partition.

2.5.2.2 Hajautusositus

Käytetään hash- funktiota taulun tai indeksin tietojen jaossa partitioihin. Soveltuu ositusmenetelmäksi, jos tietoa ei voi luontevasti jakaa ennalta määriteltyihin ryhmiin tai jos tiedon määrä partitioissa voi vaihdella suuresti. Hajautusosituken yhteydessä voidaan käyttää local -indeksejä, mutta ei global -indeksejä.

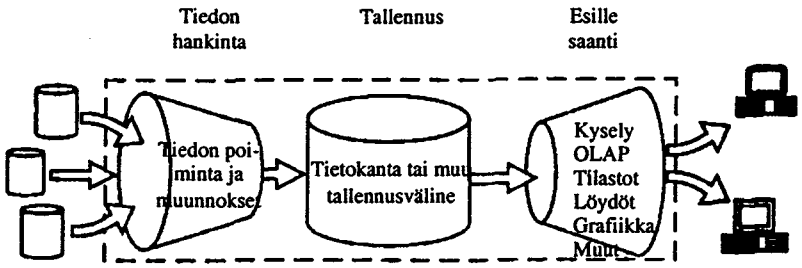
2.5.2.3 Yhdistelmäositus

Yhdistelmäositus on arvoalue- ja hajautusosituksen yhdistelmä. Tällöin taulu tai indeksi ositetaan arvoalueositusta (*range partition*) käyttäen ja näin syntyneet partitiot jaetaan edelleen alipartitioiksi käyttäen hajautusositusta (*hash partition*).

Yhdistelmäositus on looginen käsite, sillä tieto sijaitsee alipartitioissa. Partitiot ja alipartitiot ovat samassa segmentissä, minkä vuoksi niillä ei voi olla erilaiset tilamääritykset.

3 TIEDON HANKINTA

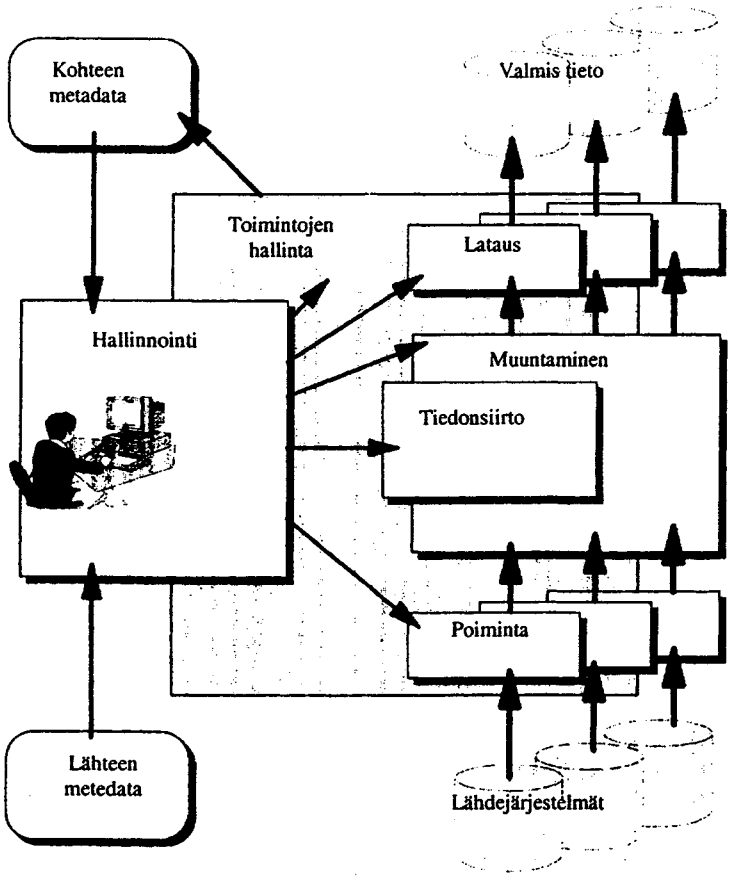
Tietovarastointi voidaan käsittää sisältävän kolme perustehtävää, mitkä erottuvat selvästi päivittäisissä rutiineissa: tiedon hankinta (*acquisition, populating*), tallennuksen (*storage*) ja tiedon esillesaannin (*access*) (Mattison, 1996, s. 10, Mattison, 1997, s. 80 ja Mattison, 1999, s.153), kuten kuvio 5 osoittaakin. Tässä tutkielmassa on tarkoitus keskittyä tiedon hankintaan.



Kuvio 5. Tietovaraston kolme komponenttia (Mattison, 1999, s.153)

Tiedon hankinnan katsotaan sisältävän tiedon poiminnan (*Extraction, capture*), muuntaminen (*Transformation*) sekä siirron ja latauksen tietovarastoon (*Transportation, apply*), minkä mukaan sitä nimitetään ETT-prosessiksi. Kuviossa 6 voidaan nähdä nämä tietovarastoinnin kolme toiminnallista komponenttia. Tämä prosessi on useimmiten eräajotyyppistä ja kerran vuorokaudessa tapahtuvaa, mutta voi olla myös jatkuvaa tosiaikaista toimintaa. Tätä on aina pidetty teknisesti haastavana tietovaraston rakentamisen osana ja siksi siihen on kehitettykin välineitä, jotka helpottavat tätä toimenpidettä. Siltikin tiedon poiminta operatiivisista järjestelmistä, yhtenäistäminen, puhdistus ja tallennus tietovarastoon on kallein ja aikaa vievin osa tietovaraston rakentamista (Devlin, 1997, s.177) sekä määrittelyvaiheessa että myös käytön aikana.

Tiedon hankinnassa tulee huomioida seuraavat seikat. Operatiiviset järjestelmät sisältävät harvoin historiatietoa tapahtumista, vaikka lähtötiedot ovatkin tavallisesti tiettyyn ajanhetkeen tai ainakin aikaväliin sidottuja. Lähtötiedot operatiivisissa järjestelmissä ovat usein hajanaisia ja voivat olla huonosti rakenteellisia. Lisäksi tietovaraston täytyy tallettaa täydelliset historiatiedot, vaikka lähtötiedot olisivatkin jaksottaisia. Operatiivisten järjestelmien ominaisuudet määräävät sen, miten tiedot voidaan toimittaa tietovarastoon. Lähtötietojen hetkellisyys määrää tiedon hankinnan tavat, joilla muuttunut tieto päivitetään tietovarastoon. Kaiken kaikkiaan sekä lähde- että vastaanottavan järjestelmän suhteet ovat monimutkaiset. Ne saattavat vaatia niin pitkälle vietyjä tiedon muunnostoimintoja, että mitkään valmiit muunnostoiminnot eivät riitä, vaan ne on tehtävä räätälöimällä.



Kuvio 6. Tietovaraston toiminnalliset komponentit (Devlin,1997 ,s.122)

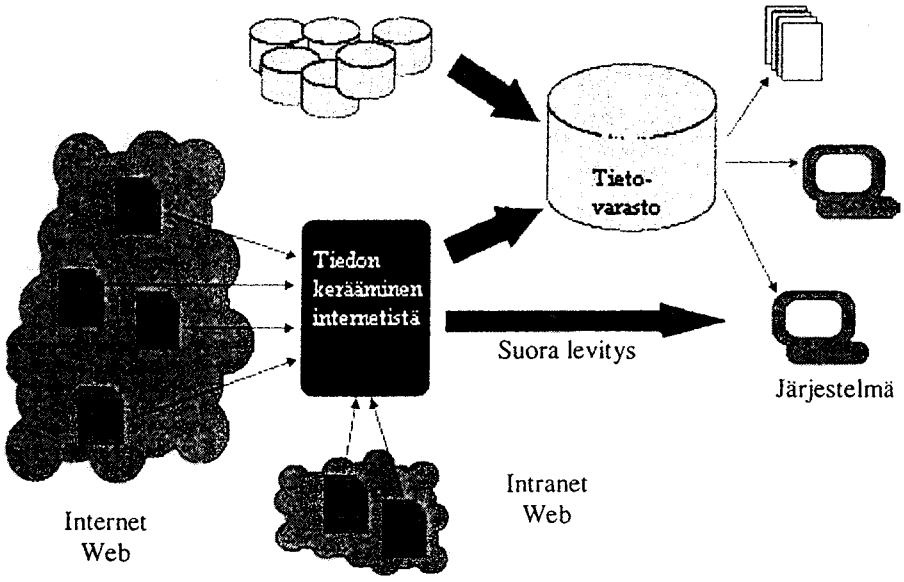
3.1 Sisäiset ja ulkoiset lähdejärjestelmät

Tiedon hankinnalla tarkoitetaan tiedon keräämistä organisaation eri paikoista tallennettavaksi tietovarastoon. Normaalisti ne jaetaan sisäisiin ja ulkoisiin lähdejärjestelmiin (Kelly, 1996, s.32). Sisäiset lähdejärjestelmät ovat yrityksen omista operatiivista tietojärjestelmistä kerättävää tietoa, mikä on luonnollisesti perinteisin ja suurin tietojen lähde. Nykyisin kuitenkin tietovaraston rakentamisessa yhtenä tärkeänä tavoitteena on pyrkiä liittämään myös yrityksen ulkopuolisen maailman tapahtumat tietovaraston avulla yrityksen sisäisällä tehtäviin päätöksiin.

Tietovarastoinnin lähtökohtana on koota tietoa organisaation operatiivisista järjestelmistä, joita aikojen kuluessa on kertynyt mahdollisesti lukuisa määrä. Sisäisten informaatiolähteiden lisäksi organisaation ulkopuoliset tiedon lähteet ovat tuoneet mukanaan uuden haasteen tiedon hankkimiselle: liiketoiminnan kansainvälistyminen ja yhteistoiminnan lisääntyminen yrityksen sidosryhmien kuten alihankkijoiden, tavarantoimittajien ja -hankkijoiden välillä, vaatii huomioimaan myös nämä tarpeet. Kiristyvä kilpailu vaatii saumatonta yhteistyötä ja informaation saantia myös organisaation ulkopuolelta. Ulkoisen tiedon analysoinnin tarkoituksena on havaita nopeasti muuttuvan toimintaympäristön antamat mahdollisuudet, tunnistaa mahdolliset uhkatekijät ja synergiaedut (Kelly, 1996, s.32).

Internet on tuonut mukanaan uuden mahdollisuuden saada helposti tietoa sähköisessä muodossa. Tässä yhteydessä onkin alettu puhua termistä *Web Farming, tiedon kerääminen internetistä.* Sillä tarkoitetaan informaation keräämistä erilaisilla välineillä internetistä ja tallettamista tietovarastoon. Tämä on valtava ja haastava tiedon lähde, joka vaatii hyvää suunnittelua antaakseen halutun tuloksen. Hackathorn (1999, http://www.webfarming.com/) toteaakin, että ulkopuolisen tiedon käyttö on välttämätöntä, jotta liiketoimintaa voidaan ohjata ja muuttaa vastaamaan ulkopuolisen maailman muuttuviin haasteisiin. Muutoksia on ehdottoman tärkeää seurata, sillä se mikä muuttuu ei ole yritys itse vaan sen ympäristö. Jos sitä ei seuraa, yritys ei pysy silloin kehityksen mukana ja kilpailijat saavat etu-

lyöntiaseman. Mattison (1999, s.8) määrittelee tiedon web-käytön perusteella termin *Web Warehousing*, ratkaisumallina rakentaa tietokonejärjestelmä, minkä pääasiallisena tehtävänä on tunnistaa, luetteloida, noutaa, tallettaa ja analysoida informaatiota käyttämällä web-teknologiaa. Kuvio 7 kuvaa Hackathornin käsitystä tiedon keruusta internetistä.



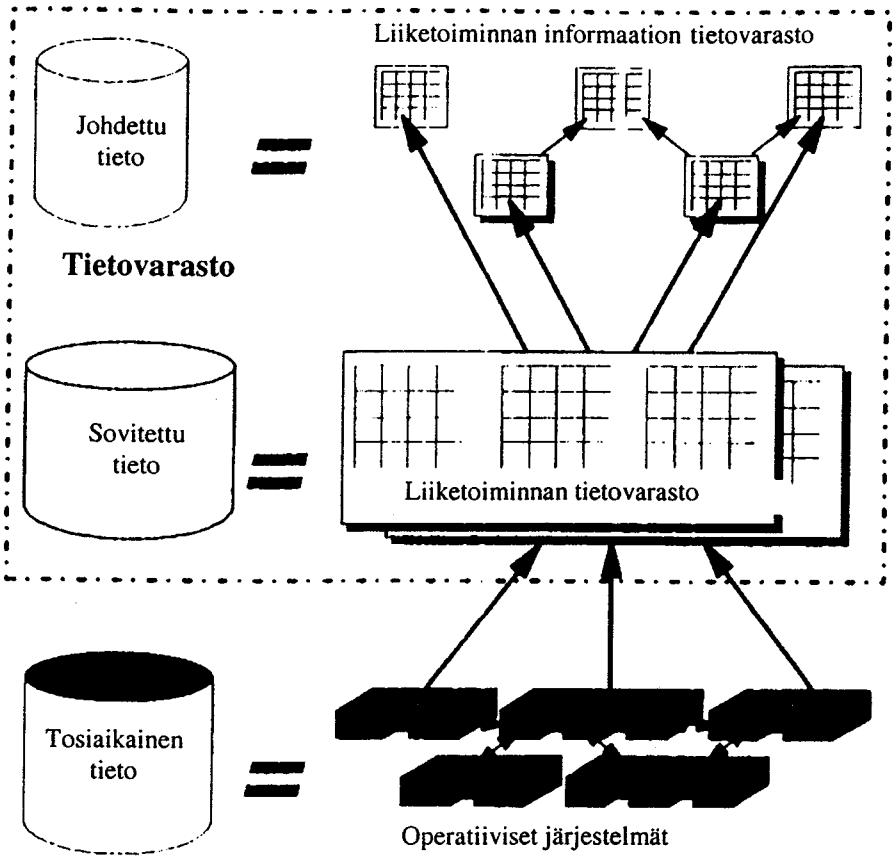
Kuvio 7. Tiedon kerääminen internetistä. Viitattu 26.9.1999. Saatavilla muodossa: <URL:http://www.webfarming.com/>

Tietojen keruu internetistä on vielä kuitenkin alkutekijöissään. Tietomäärä on valtava ja suurin ongelma onkin halutun tiedon suodattaminen valtavasta tietomäärästä. Toisaalta tietojen keruu täytyy olla automatisoitua. Tietojen keräämiseksi internetistä tarvitaan monenlaisia välineitä ja ohjelmistoja. Mikään yksittäinen tuote ei pysty tekemään sitä yksin. Hackathorn (1999, s. 182) luettelee tiedon hakuun käytettäviä välineitä. Tällaisia ovat internet-selaimet, web-agentit, jotka seuraavat kohdesivuilla tapahtuvia muutoksia ja noutavat muuttuneita tietoja. Analyysi- ja muunnostyökaluina voidaan käyttää hakukoneita. Myös tiedon visualisointiin liittyvät työvälineet, joita esim. Helsingin yliopiston tutkijaryhmä on kehittänyt Websom -projektissaan. Lisäksi laajennetut relaatiotietokannat ja pai-

kalliset tietovarastot sekä tietämyksen hallintajärjestelmät voivat olla tiedon keruun välineinä.

3.2 Tietovarastoon tallennettavan tiedon kolme tyyppiä

Liiketoiminnassa käytettävä tieto (*business data*) on informaatiota, jota tarvitaan liiketoiminnan pyörittämiseen ja hallitsemiseen. (Devlin, 1997, s.44). Se kuvaa liiketoiminnan toimintoja ja reaalimaailman käsitteitä. Liiketoimintatieto voidaan Devlinin mukaan luokitella sen käyttötarkoituksen (operational, informational), ulottuvuuden (yksityiskohtainen, summattu), käyttötyypin (luku-kirjoitus/vainluku) ja voimassaolon (ajankohtainen, tietynhetkinen, määräaikainen) mukaan. Liiketoimintatieto voidaan edelleen jakaa kolmeen tyyppiin, jotka ovat pohjana Devlinin teorialle tietovarastoinnin eri arkkitehtuuritasoille (Devlin, 1996, s.46-50). Kuvion 8 mukaan nämä tasot ovat tosiaikainen tieto (*real-time data*), johdettu tieto (*derived data*) ja sovitettu tieto (*reconciled data*). Kimballin mukaan tämä kolmijakoinen arkkitehtuuri ja sen tasot ovat käsitteellisiä, mutta niille voidaan kuitenkin löytää fyysiset ilmentymätkin, jotka tosin eivät ole koskaan yhdessä paikassa ja ovat toteutettu useilla eri tavoilla (Devlin, 1997, s.126). Tiedon rakenteen näkökulmasta katsottuna nämä kolme arkkitehtuuritasoa voidaan nimetä seuraavasti. Tosiaikaista tietoa vastaa operatiivisten järjestelmien tieto. Itse tietovarastoon katsotaan kuuluvaksi johdettu tieto ja sovellettu tieto. Sovellettu tieto on sitä tietoa, joka on kerätty useista eri operatiivisten järjestelmien tiedoista ja yhdistetty johdonmukaiseksi kokonaisuudeksi. Tätä tasoa voidaan kuvata termillä liiketoiminnan tietovarasto (*business data warehouse*). Se sisältää ne tietovaraston taulut, joihin tietovaraston informaatio on fyysisesti tallennettukin. Ylin taso eli johdettu tieto ei välttämättä ole tallennettu fyysisesti mihinkään. Tätä tasoa kuvataan termillä liiketoiminnan informaation tietovarasto (*business information warehouse, BIW*). Tämä on se taso, jota käyttäjät pääasiassa hyödyntävät käyttäessään tietovarastoa. Se voi sisältää tietovaraston tauluista muodostetut kuutiot, joita hyödynnetään moniulotteisen tiedon haussa.



Kuvio 8. Kolmitasoinen arkkitehtuuri ja tietovarasto (Devlin, 1997, s. 127)

Tiedon alin taso on tosiaikainen tieto (*real-time data*), joka on operatiivisen järjestelmän käyttämää tietoa. Sen on luonut ja sitä ylläpitää joko operatiivinen järjestelmä tai joku muu tuotantojärjestelmän sovellus. Usein se on suurkonejärjestelmien tiedostojen tai tietokantojen sisältämää hyvin tärkeää tietoa, jonka tarkoituksena on liiketoiminnan pyörittäminen. Tällaista tietoa käytetään asiakastietojen hallinnassa, laskutuksessa, varastonvalvonnassa ja tuotannonohjauksessa.

Tiedon toinen taso on johdettu tieto (*derived data*). Se on yksinkertaisesti tietoa, joka on jonkun prosessin läpi johdettua tosiaikaista tietoa. Se on luonteeltaan enemmänkin vain lukukäyttöön (*read-only*) tarkoitettua liiketoiminnan hallintaan keskittyvää tietoa tietystä ajanhetkestä tai -jaksosta, eikä jokapäiväiseen ylläpitotoimintoihin tarkoitettua kuten tosiaikainen tieto on. Johdettu tieto voi olla joko

yksityiskohtaista tai summattua tietoa, joka on perinteisesti tarkoitettu päätöksenteon tueksi. Nykyisin se sijoittuu organisaatiossa kaikkialle aina suurkonetietojärjestelmistä henkilökohtaisien tietokoneiden väliin. Johdetun tiedon laajuus vaihtelee liiketoiminnan vaatimusten mukaan: yksinkertaisimmillaan se voi olla otos yksityiskohtaisesta tiedosta tai summattua tietoa, joka on yhdistetty korkeamman tason kokonaisuudeksi. Kummassakin tapauksessa se voi olla esimerkiksi vain muutaman kentän otos yksityiskohtaisesta tiedosta. Tämän lisäksi se voi olla myös useamman eri tiedon yhdistämistä niin, että saatu tieto on yhdessä enemmän kuin eri tiedonjyvät erillään. Tätä sanotaan tiedon rikastamiseksi (*enrichment*) ja se on johdonmukaista ainoastaan silloin, kun yhdistetyt tiedot liittyvät loogisesti toisiinsa. Kun nämä tiedot saadaan liitettyä selkeästi toisiinsa, päädytään kolmanteen tiedon tasoon eli sovitettuun tietoon (*reconciled data*).

Sovitettu tieto luodaan toiminnoilla, jotka on suunniteltu varmistamaan saadun tiedon sisäinen yhdenmukaisuus. Tämä toiminto käsittelee tosiaikaista tietoa yksityiskohtaisella tasolla. Toinen sovitetun tiedon tuottamisen näkökanta on se, että se ylläpitää tai luo ajallista ulottuvuutta olevaa tietoa. Siksi sovitettu tieto voidaan nähdä johdetun tiedon yhtenä luokkana. Perinteisesti sovitettua tietoa ei ole nimenomaan määritelty. Useimmissa tapauksissa sitä ei edes ole fyysisesti talletettu mihinkään vaan se on looginen tulos joistakin toimenpiteistä, jotka tehdään sovitetun tiedon luontiprosessissa. Tieto on tällöin talletettuna enintään väliaikaisiin tiedostoihin. Sovitettu tieto on tietovaraston keskeinen elementti. Tosiaikainen tieto ei ole itsestään johdonmukaista tietojärjestelmien välillä; tämä tekee sovelletun tiedon välttämättömäksi. Koska tietoa poimitaan useista eri lähteistä ja yhdistetään, tiedon rakenne ja asiayhteys täytyy ensin analysoida, jotta tiedon yhdistämisen säännöt voidaan määritellä. Tyypillisesti tämä edellyttää eri lähteistä tulevien tietojen yhteensovittamista, muuttamista ja konvertointia johdonmukaiseksi kokonaisuudeksi. Joskus se vaatii myös tietosisällön korjaamista. Yleisiä ongelmia ovat yksikäsitteisten avaimien poikkeavuudet eri tietojärjestelmissä. Esimerkkinä tästä voi olla asiakas- tai tuotenumeron pituuden vaihtelut. Kokonaan eri tyyppiset koodaukset samaa tarkoittavassa avainrakenteessa on tavallista. Tällöin manuaalisesti rakennettu avainten vastaavuustaulu on ainoa

vaihtoehto. Muita tarvittavia muunnoksia ovat myös tietokenttien erilaiset merkintätavat: sukupuolta osoittamaan saatetaan käyttää kirjaimia (M/N) tai numeroita (0/1). Vielä mutkikkaamman ongelman muodostavat erilaisten tietojärjestelmien ajallinen ulottuvuus. Operatiivisissa järjestelmissä tieto on usein tallennettu päivän tai jopa täsmällisen kellonajan tarkkuudella, kun taas toinen järjestelmä summaa tapahtumat kahden viikon, tilikauden, kuukauden, vuosineljänneksen tai puolen vuoden palasiksi. Taloushallinnon järjestelmä saattaa ylläpitää informaatiota ainoastaan kuukauden vaihteen tilanteesta. Näiden eri järjestelmien yhteensovittaminen on vaativaa.

3.3 Tiedon poiminta

Tiedon poiminnan voidaan käsittää sisältävän tiedon tunnistamisen, tarpeellisuuden ja käytettävyyden analysoinnin sekä tiedon keräämisen operatiivisista tai muista ulkoisista lähdejärjestelmistä (extraction, capture). Tämän lisäksi saatava tieto on käsiteltävä sellaiseksi, että se voidaan ajatella tallennettavaksi tietovarastoon. Vaikka lähdejärjestelmästä voitaisiinkin tuoda kaikki tieto tietovarastoon ja sitten hylätä tarpeeton, on paljon tehokkaampaa tuoda vain tarvittava osuus (Devlin, 1997, s.179). Devlin erottaa tiedon tunnistamisesta kaksi tapaa: muuttumaton poiminta (*static capture*) kokoaa sellaista tietoa lähdejärjestelmästä, millä ei ole ajallista kiinnitystä. Tämän lisäsi tietokannat sisältävät tietoa, joka muuttuu ajan kuluessa. Tällaisesta tiedosta pitää usein tallettaa myös muutosten eri historiatasot. Joissakin tapauksissa muuttumaton poiminta säännöllisin väliajoin on riittävää, mutta poiminnan tehokkuus ja tarve muuntaa hetkellistä tai osittain jaksottaista tietoa vaativat toisenlaistakin käsittelyä. Tämän tyyppistä tiedon poimintaa sanotaan lisääväksi poiminnaksi (*incremental capture*).

Muuttumaton poiminta tarkoittaa käytännössä sitä, että lähdejärjestelmästä otetaan otos (*snapshot*) tiettynä ajanhetkenä. Otos saattaa sisältää kaikki lähdejärjes-

telmän tiedot, mutta tavallisesti siihen poimitaan vain tarkkaan määritellyt tiedot. Muuttumatonta poimintaa käytetään usein, vaikkakaan se ei yhtä yleinen kuin lisäävä poiminta. Sitä käytetäänkin esimerkiksi silloin, kun operatiivisesta järjestelmästä poimitaan tietovaraston alkutilanne (*alkulataus*). Toinen mahdollinen käyttötilanne on silloin, kun operatiivinen järjestelmä sisältää ja ylläpitää koko tiedontarpeen historiatiedot ja samalla tiedon määrä on rajallinen (Devlin, 1997, s.179).

Lisäävä poiminta perustuu siihen, että tiedolla on lähes aina ajallinen ulottuvuus ja siksi se pitääkin käsitellä niin, että aikaulottuvuus otetaan huomioon tietovarastoon tallennettaessa. Muuttuneen tiedon määrä on suhteellisen pieni verrattuna tarvittavan tiedon kokonaismäärään. Siksi lisäävä poiminta, joka kohdistuu muuttuneeseen tietomassaan on tehokkaampaa kuin muuttumaton poiminta. Poiminnan on kerättävä muuttunut tieto niin, että lähdejärjestelmän muutokset siirtyvät tietovarastoon muodostaen sinne järkevän ja oikean kuvan tapahtuneista muutoksista. Luonnollisesti lisäävä poiminta on paljon monimutkaisempi kuin muuttumaton poiminta. Tietojen ajallisen ulottuvuuden erilaiset käsittelytavat ja erilaiset teknologiat lähdejärjestelmissä verrattuna tietovaraston vastaaviin vaikuttavat siihen, että lisäävällä poiminnalla on lukemattomia erilaisia variaatioita. (Devlin, 1997, s.179-180).

3.3.1 Tiedon tunnistaminen

Tietovarastoon haluttava tieto pitää poimia operatiivisista järjestelmistä, varmuuskopioilta, tai muista organisaation sisäisistä tai ulkoisista lähdejärjestelmistä. Tämä vaatii paljon yhteistyötä eri osapuolien välillä, sillä se on vaativa tehtävä (Mattison, 1996, s. 133). Käytössä olevat järjestelmät voivat olla useiden eri ohjelmistotoimittajien käsialaa, mikä mutkistaa olemassa olevan tiedon kartoitusta ja halutun tiedon määrittelyä. Usein vuosikymmenten aikana kehitetyt ja

organisaation tarpeiden mukaan eläneet järjestelmät ovat puutteellisesti dokumentoituja. Dokumentit ovat vanhentuneet ja tieto siitä, mitä tietoja järjestelmä todella sisältää ja miten ne saadaan sieltä ulos, on monesti muutaman järjestelmää ylläpitävän henkilön muistinvaraisena tietona.

3.3.2 Tietosisällön arviointi

Kun tieto on löydetty, on varmistuttava siitä, että se on todellisuutta vastaavaa, asianmukaista ja käyttökelpoista. Monissa järjestelmissä tietoa on tallennettu vapaisiin kenttiin niin, että sisältö ei vastaakaan kentän kuvausta. Tietosisältö saattaa olla myös muuhun asiayhteyteen sidottua ja sen järkevyys täytyy arvioida tiedon käyttäjien kanssa.

Tiedon eheys on varmistettava. Tämä koostuu seuraavista vaiheista:

- varmistutaan tiedon lähteen alkuperäisyys (*identity-integrity*)
- varmistutaan tiedon tallennustavan oikeellisuus (*population-integrity*)
- kehitetään keinot oikaista löydetyt virheet
- tunnistetaan ohjelmat ja tiedostot, joita käytetään tallennuksessa

(Mattison, 1996, s. 134).

Tiedon lähteen alkuperäisyydellä tarkoitetaan sitä, vastaako käsiteltävä tieto sitä, mitä käyttäjä siinä otaksuu olevan. Tiedon sisältö on määriteltävä niin tarkalla ja selvällä tasolla, että organisaatiossa ei tule tulkintaepäselvyyksiä sen sisällöstä. Tallennustavan oikeellisuus merkitsee sitä, että tiedot pitää olla tallennettuna sovitulla tavalla, joka on hyödynnettävissä helposti. Esimerkiksi päivämäärä voi olla tallennettuna numeeriseen, merkkimuotoiseen tai päivämääräkenttään. Niiden yksiselitteinen käyttö on vaikeaa. Toinen ongelma on epätäydellisesti tallennetut tietueet, joiden puuttuvat tiedot aiheuttavat suuria ongelmia siirrettäessä tietoja tietovarastoon.

Erillisten lähtötiedostojen yksittäisten eheystarkastuksen lisäksi erilaisista lähtöjärjestelmistä tulevien tietomassojen tietosisällön yhdenmukaistaminen on tärkeä tehtävä. Se koostuu Mattisonin (Mattison, 1996, s. 137) mukaan kahdesta pääluokasta: viite-eheydestä (*referential integrity*) ja aikatekijöistä (*timing issues*).

Viite-eheydellä tarkoitetaan tietoelementin suhdetta toisiin tietoelementteihin. On olemassa kolmenlaisia viite-eheysvaatimuksia, jotka tulee ottaa huomioon: käytettävyyden eheys (*utilization integrity*), tallennustavan yhtenäistäminen (*population standardization*) ja sisäinen yhtenäistäminen (*interfile standardization*). Käytettävyyden eheys tarkoittaa esimerkiksi sitä, että rahayksikkö tallennetaan kaikkialla aina samalla tarkkuudella pyöristämättä sitä. Tallennustavan yhtenäistäminen vaatii tallentamaan samaa tarkoittavat tiedot aina samalla tavalla; esimerkiksi saman kadun nimi ei saa löytyä usealla eri tavoilla tallennettuna. Sisäinen yhtenäistäminen vaatii, että sama tieto tallennetaan kaikista eri lähteistä samalla tavalla, jotta saatu tieto olisi yhteismitallista.

Aikatekijöiden ja historiatiedon huomiointi on yksi tietovarastoinnin tärkeimmistä asioista. Vaikka lähtötiedoilla ei olisikaan varsinaisesti ajallista ulottuvuutta, niille joudutaan tietovarastoon tallennuksen yhteydessä sellainen antamaan. Monilla operatiivisten järjestelmien lähtötiedoilla on jo valmiina ajallinen ulottuvuus, mutta se poikkeaa järjestelmästä riippuen. Näiden yhteensovittaminen vaatii sekä lähdejärjestelmän että tietovaraston vaatimusten perinpohjaisen tuntemisen. Lisäksi samalla on otettava kantaa siihen, miten usein tiedot kannattaa tallettaa tietovarastoon (Mattison, 1996, s. 140).

3.3.3 Tiedon poimimisen operatiivisista järjestelmistä

Tiedon poiminta operatiivisista järjestelmistä voi tapahtua monella eri tavalla riippuen arkkitehtuurista. Jos tiedot joudutaan siirtämään laiteympäristöstä toi-

seen, siirto joudutaan usein tekemään ajoitettuina tiedostosiirtoina. Sama laiteympäristö ja arkkitehtuuri antaa enemmän automatisoituja mahdollisuuksia.

Tiedon poiminta voidaan jakaa muuttumattomaan, heti tapahtuvaan ja viivästettyyn poimintaan. Muuttumaton on yksinkertaisin ja siinä poimitaan halutuista tiedoista otos, joka siirretään tietovarastoon (Devlin, 1997, s.182).

Usein tietojen poiminta ei kuitenkaan ole yksinkertaista johtuen siitä, että tiedot muuttuvat alati. Tällöin myös tietojen muutokset olisi pystyttävä toimittamaan jo niiden tapahtumahetkellä tietovarastoon, sillä muuttuneen tiedon erottaminen aikaisemmasta tilasta myöhemmin voi olla vaikeaa tai jopa mahdotonta, sillä muutoshistoriaa ei kovinkaan usein tallenneta operatiivisiin järjestelmiin. Jos kaikki operatiivisen järjestelmän tietojen muutokset pitää tallentaa tietovarastoon, voidaan käyttää apuna sovellusta, liipaisimia (*triggereitä*) tai tapahtumalokeja.

Jos peräkkäisien tapahtumien tallentaminen ei ole mahdollista tai antaa virheellisen lopputuloksen, joudutaan käyttämään viivästettyä tiedon poimintaa. Tällainen voi perustua aikaleimoihin tai tiedostojen vertailuun.

Inmon käsittelee tietojen poimintaa hieman ristiriitaisesti (vrt. Inmon, 1996, s.77 ja s.195). Hänen mukaansa viivästetty lokitietoihin perustuva tietojen poiminta on tehokas, mutta vaatii melkoista asiantuntemusta. Tämän vuoksi hän ei suosittele sitä. Kuitenkin myöhemmin hän toteaa, että se on itse asiassa käyttökelpoisin tapa poimia tietoa tietovarastoon. Kehityspolku organisaatioissa, joihin on rakennettu tietovarasto, kulkee Inmonin mukaan seuraavasti. Aluksi tietojen poiminta tapahtuu lukemalla tiedot suoraan operatiivisesta järjestelmästä. Sen jälkeen yritetään replikointia. Lopuksi taloudellisista ja tehokkuuteen perustuvista syistä joudutaan siirtymään tietovaraston päivittämisessä muuttuneen tiedon poiminnassa (*CDC, changed data capture*) pääasiassa lokitietoihin perustuvaan poimintaan. Vain muutamat tiedostot vaativat suoran yhteyden tai replikoinnin operatiivisiin järjestelmiin päin.

3.3.3.1 Muuttumaton poiminta

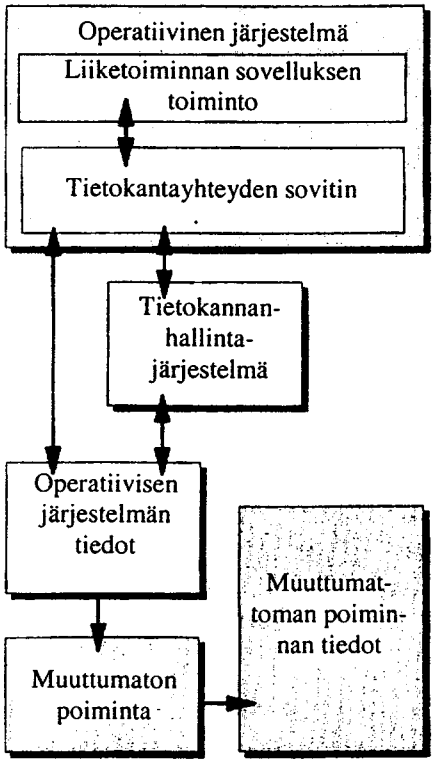
Muuttumattoman poiminnan periaatteena on ottaa otos lähtötiedoista täsmälleen siinä muodossa kuin ne esiintyvät operatiivisessa järjestelmässä. Muuttumaton poiminta käsitteleekin juuri operatiivisen järjestelmän tietoja olivatpa ne sitten tyypiltään ylläpidettäviä (esim. asiakastiedot) tai lokityyppisesti kertyviä. Tämän lisäksi ne voivat sijaita missä ja miten tahansa: joko tiedostoina tai tietokantatauluina. Ongelmana monissa organisaatioissa onkin se, että tieto on tallennettuna hyvin monella eri tavalla. Tieto voi olla talletettuna peräkkäistiedostoihin ja indeksoituihin peräkkäistiedostoihin kuten myös tietokantoihin, jotka voivat olla hierarkkisia, relaatio- tai jopa jotain muuta tyyppiä. Tämän lisäksi fyysinen talletustapa voi poiketa eri arkkitehtuureista riippuen ympäristöstä toiseen. Vaatimuksena muuttumattomalle poiminnalle on kuitenkin, että saatava tieto on oltava sovitussa ja yksinkertaisessa muodossa tietovarastoon siirtoa varten. Kuvio 9 kuvaa Devlinin näkemystä tiedon muuttumattomasta poiminnasta.

Muuttumaton poiminta voidaan jakaa kolmeen eri ulottuvuuteen (Devlin, 1997, s.183). Kohdetyyppiulottuvuus (*entity dimension*) on yksinkertaisin poimintatyyppi. Tässä tapauksessa kaikki lähdejärjestelmän tiedot poimitaan sellaisenaan rajaamatta tietomassaa kenttä- tai valintaehtotasoilla. Tätä poimintatapaa kuvaa parhaiten SQL-komento, jolla valitaan kohdetaulun kaikki kentät:

```
SELECT    *
FROM      CUSTOMER_FILE
```

Ominaisuusulottuvuus (*attribute dimension*) poimii lähdejärjestelmästä vain halutut tiedot. Tässäkään tapauksessa rajausta ei tehdä valintaehtotasolla. Tätä poimintatapaa kuvaa parhaiten SQL-komento:

```
SELECT    NAME, PHONE_NO, CITY
FROM      CUSTOMER_FILE
```



Kuvio 9. Muuttumaton poiminta (Devlin, 1997, s. 183)

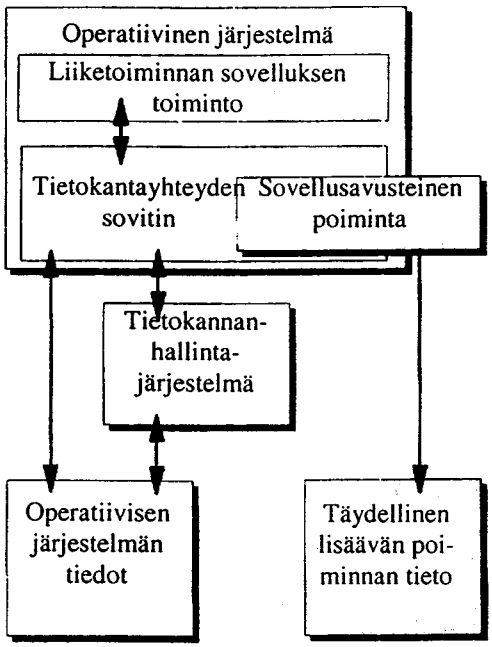
Tapahtumaulottuvuus (*occurence dimension*) poimii lähdejärjestelmästä kaikki tiedot rajaamatta tietokenttiä. Tässä tapauksessa rajaus tehdään tietueiden valintaehdolla poimimalla vain haluttu otos lähdejärjestelmän tietomassasta. Tätä poimintatapaa kuvaa parhaiten SQL-komento:

```
SELECT    *
FROM CUSTOMER_FILE
WHERE     CITY = 'Paris'
```

Luonnollisesti nämä kolme erilaista ulottuvuutta voidaan sekoittaa keskenään muodostaen näin monipuolisia poimintavaihtoehtoja.

3.3.3.2 Heti tapahtuva poiminta

Ensimmäinen lisäävän poiminnan ja heti tapahtuvan (*immediate capture*) tietojen poiminnan tavoista on sovellusavusteinen poiminta (*application-assisted capture*) (kuvio 10). Tämä on poiminnan lajeista selvimmin kaikkein välittömimmin itse tapahtuman tapahtumahetkeen sidottu. Sovellusavusteinen poiminta perustuu sii-



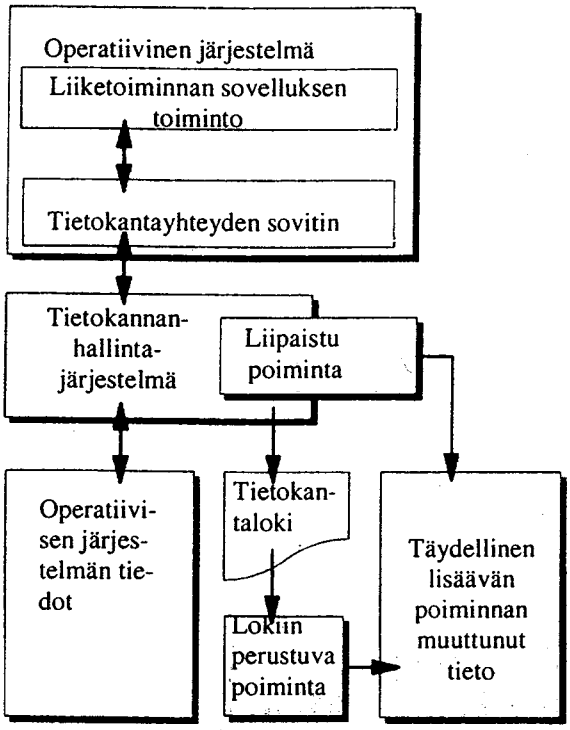
Kuvio 10. Sovellusavusteinen tiedon poiminta (Devlin, 1997, s. 184)

hen, että operatiiviseen tietojärjestelmän kiinteäksi osaksi ohjelmoidaan keskeytymätön muuttuneen tiedon kirjoitus tietueeksi rinnakkain operatiivisen järjestelmän varsinaisen tiedon ylläpidon kanssa. Tässä tapauksessa ei ole olemassa fyysisesti erillistä tunnistettavaa komponenttia, jolla tiedot kirjoitetaan vaan koodi on upotettu operatiiviseen tietojärjestelmään. Tämä tietojen poimimistapa on

hyvin joustava ja antaa täsmällisimmän lopputuloksen, mutta se on myös erittäin monimutkainen olemassa oleviin järjestelmiin nähden. Tapa vaatii sen, että operatiivinen järjestelmä tunnetaan erittäin hyvin ja että se on dokumentoitu kunnolla. Useinhan vanhat ja pitkään käytössä olevat järjestelmät ovat huonosti dokumentoituja eikä niille enää löydy ylläpitäjää. Mutta kun suunnitellaan uusia operatiivisia järjestelmiä, tätä poimintavaihtoehtoa voidaan hyödyntää parhaiten erottamalla liiketoimintalogiikka ohjelmakoodista: luodaan yleiskäyttöisiä komponentteja, joille on määritelty tarkkaan ohjelmointirajapinta (*API, application programming interface*) hoitamaan tietokantayhteyttä ja samalla myös ohjelmistoavusteista poimintaa. Tällä tavalla on useita hyötynäkökohtia: järjestelmäsuunnittelijat voivat keskittyä liiketoimintalogiikan kehittämiseen tietokannan käsittelylogiikan sijaan. Lisäksi jos halutaan vaihtaa tietokannanhallintajärjestelmä, kaikki siihen liittyvä logiikka on erillään sovellusohjelmistosta eikä aiheuta muutoksia sinne. Samalla tavalla sovellusavusteinen tietojen poiminta on erillään itse järjestelmästä, ja muutokset eivät aiheuta koodimuunnoksia itse järjestelmään. Tämä menetelmä ei ole kuitenkaan kovin helppo toteuttaa. Varsinkin jos muutosten päivityksen hoitaa tietokannanhallintajärjestelmä automaattisesti, saattaa tilanne ennen ja jälkeen muutoksen olla vaikea saada käsiteltäväksi. Lisäksi tietueella saattaa olla satoja tietoja, joista vain osaa on muutettu. Näiden muuttuneiden osien tunnistus on työlästä. Vastaavanlainen ongelma syntyy, kun operatiivisesta järjestelmästä poistetaan tietue fyysisesti. Tällainen tapahtuma tulee kuitenkin välittää tietovarastoon joko tietyn tietueen voimassaolon päättymistietona, tai tietueen voimassaolotilan päättymisenä. (Devlin, 1997, s.184-186).

Toinen lisäävän poiminnan ja heti tapahtuvan (*immediate capture*) tietojen poiminnan tavoista on liipaistu poiminta (*triggered capture*) (kuvio11). Liipaisu tapahtuu tietokannanhallintajärjestelmän toimesta. Liipaisu tapahtuu silloin, kun määritelty tapahtuma (*event*) aktivoi halutun toimenpiteen seurauksena tietokantaan tallennetun proseduurin. Tapahtuma aikaansaa toisen transaktion, joka tekee siinä määritellyn päivityksen esimerkiksi toiseen tietokantatauluun. Se mitä liipaisun käynnistävä tapahtuma kirjoittaa, on määriteltävä tilanteen mukaan. Tietyn ken-

tän päivitys saattaa vaatia joko muutoksen tai muutoksen jälkeisen tilan tallennuksen toiseen tauluun tietovarastoon tallentamista varten. Tällainen toimenpide soveltuu paremmin tietokantakäsittelyyn kuin tiedostojen käsittelyyn. Toteutukset vaihtelevat tietokannanhallintajärjestelmästä riippuen toisistaan.



Kuvio 11. Liipaistu ja lokiin perustuva poiminta (Devlin, 1997, s. 186)

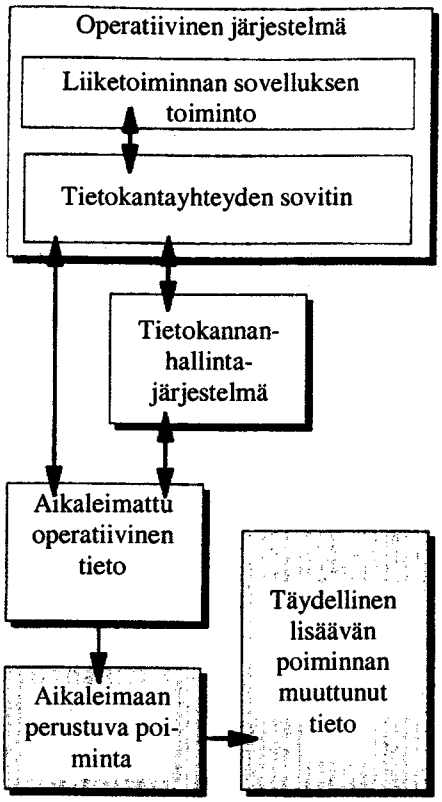
Ongelmana tässä tapauksessa on se, että tapahtuma synnyttää toisen tapahtuman, mikä luonnollisesti vie järjestelmäresursseja päivitettäessä suuria määriä tietueita (Devlin, 1997, s.187).

Kolmas lisäävän poiminnan ja heti tapahtuvan (*immediate capture*) tietojen poiminnan tavoista on lokiin perustuva poiminta (*log capture*). Se perustuu tietokannanhallintajärjestelmän itsensä tekemiin päivityksiin, kuten liipaistu poimintakin. Useimmissa tapauksissa tietokannanhallintajärjestelmä ylläpitää tietokantatau-lujen muutoshistoriaa lokitiedostoissa (*archaive logs*), joita saattaa olla useita toisiinsa ketjutettuina. Näitä lokitiedostoja käytetään tarvittaessa järjestelmän elpymiseen (*recovery*) virhetilanteiden yhteydessä. Näitä lokitiedostoja voidaan samalla

käyttää myös lokiin perustuvaan tietojen poimintaan tietovarastosovelluksen yhteydessä. Edellytyksenä on luonnollisesti se, että ominaisuus on kytketty päälle tietokannanhallintajärjestelmässä ja että lokit ovat riittävän pitkiä. Lokitiedostot ovat usein uudelleenkäytettäviä, jolloin vanhin loki otetaan uudelleen käyttöön, kun viimeinen loki on kirjoitettu täyteen. Lokiin perustuva poiminta on tavallisesti hyvin tehokas tapa, sillä se on hyvin optimoitu tietokannanhallintajärjestelmän osa, joka tehdään joka tapauksessa tietokannan eheyden varmistamiseksi. Lokitietojen analysointi, siirtäminen ja kirjoitus tietovarastoon voidaan tehdä sellaisena aikana, että se ei häiritse operatiivista järjestelmää. Usein lokitietojen käsittely tehdäänkin jopa eri koneessa kuin operatiivinen järjestelmä toimii. Lokitietojen analysointi ja käsittely ei kuitenkaan ole kovin yksinkertaista: lokilla on jokainen päivitys, joka tehdään operatiivisen järjestelmän tietokannassa. Tämän vuoksi loki sisältää valtavan määrän tietoa, josta vain tarvittava osa on pystyttävä poimimaan. Tietojen esille saanti ja valinta on vaativa tehtävä. Myös tietokannan eri versioiden väliset erot saattavat muuttaa ja monimutkaistaa lokitietojen muotoa ja käsittelyä (Devlin, 1997, s.187-188). Inmonin mukaan (Inmon,1996, s.77) lokin tulkinta saattaa vaatia melkoista asiantuntemusta ja lisäksi se sisältää aivan liikaa informaatiota, joten sen käyttö on vaikeaa. Lokitietojen käytön sijaan Inmon suositteleekin ns. muutostiedostojen (*delta-file*) käyttöä. Muutostiedostoon kirjoitetaan operatiivisesta järjestelmästä ne tiedot, joihin on tehty muutoksia. Tämä tapa rajaa latausprosessissa läpikäytävien tietojen määrän minimiin. Tämä poimintatapa vaatii muutoksia operatiiviseen järjestelmään, jotta se osaisi kirjoittaa muutostiedot tiedostoon. Siksi se voidaan ainakin osittain katsoa olevan Devlinin jaottelussa sovellusavusteista poimintaa.

3.3.3.3 Viivästetty poiminta

Viivästetyn poiminnan ensimmäinen tyyppi on aikaleimaan perustuva poiminta (*timestamp-based capture*) (kuvio 12). Se on itse asiassa muuttumattoman poi-

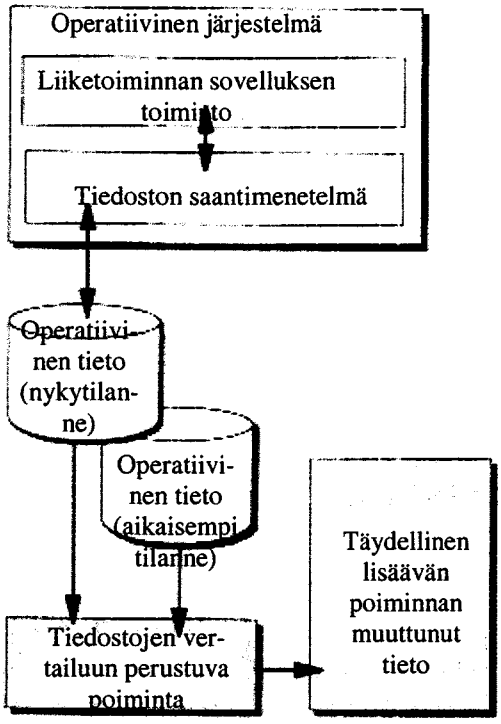


Kuvio 12. Aikaleimaan perustuva poiminta (Devlin, 1997, s. 188)

minnan yksi tapaus. Tässä tapauksessa lähtötiedoissa täytyy olla yksi tai useampi kenttä aikaleimatietoa varten, jota sitten voidaan käyttää tietueen valitsemiseen poiminta-ajossa. Poiminta-ajo tapahtuu tietyin väliajoin ja sen tehtävänä on ottaa mukaan edellisen ajon jälkeen muuttuneet tiedot perustuen juuri tietueelle tallennettuun aikaleimaan. Tämä tekee tästä poiminnan tavasta selvästi viivästetyn, sillä tiedot poimitaan aina tietyn viiveen jälkeen niiden syntymishetkestä. Ongelman muodostavat operatiivisen järjestelmän ne tiedot, jotka ovat hetkellisiä. Tietueeseen saattaa kohdistua useampia päivityksiä sen aikavälin aikana, jona tiedot poimitaan tietovarastoon. Tässä tapauksessa aikaleiman mukaan saadaan poimittua vain viimeisin tila. Tietojen luonteesta riippuen tässä tapauksessa ehkä menetetään aikaisemmat muutokset. Jos tiedon muutoksen kaikki tilat halutaan poimia tietovarastoon, on operatiivisen järjestelmän pystyttävä tallentamaan tiedon kaikki eri versiot aikaleimoineen poimintaa varten. Vastaavanlainen ongelma syntyy tietojen poiston yhteydessä. Tieto poistettavasta tietueesta on pystyttävä jättämään järjestelmään niin, että poiminta-ajo voi siirtää myös tällaisen infor-

maation tietovarastoon. Tämä vaatii operatiiviselta järjestelmältä erillisiä toimia, jotta tietueen fyysistä poistoa ei tehtäisikään (Devlin, 1997, s.188-190). Ongelmaksi tässä poimintatavassa saattaa kuitenkin tulla se, että lähtötiedot ovat monesti sattumanvaraisesti aikaleimalla varustettuja (Inmon,1996, s.77).

Viivästetyn poiminnan toinen muoto on tiedoston vertailuun perustuva poiminta (*file comparison*) (kuvio 13). Tätä poimintatapaa ei yleensä käytetä kuin niissä tapauksissa, joissa mikään muu poiminta ei onnistu. Tiedoston vertailuun perustuva tapa edellyttää, että tietokannasta tai sen tiedoista tehdystä tiedostosta on ole-



Kuvio 13. Tiedoston vertailuun perustuva poiminta (Devlin, 1997, s. 190)

massa kaksi eri ajanhetken kopiota: yksi aikaisemmasta ajanhetkestä ja yksi nykytilanteesta. Näitä kahta eri ajanhetken tilannetta vertaillaan ja etsitään erot, jotka sitten poimitaan siirrettäväksi tietovarastoon.

Tämä vaatii paljon resursseja. Eri tilannetiedot vievät paljon levytilaa ja vertailusekä lajitteluohjelmat vievät runsaasti koneresursseja suhteellisesti pienen muutettujen tietueiden määrän löytämiseksi. Tälläkin tavalla on hyvät puolensakin. Jos vertailtavien tietojen määrä on rajoitettu eikä mitkään muut tavat sovellu ja kyseessä on erityisesti tiedostoihin pohjautuva järjestelmä, on tämä tapa mahdollisesti käyttökelpoinen. Silloin itse järjestelmiin ei tarvitse tehdä muutoksia. Tätä poimintatapaa kannattaa käyttää yhdessä muuttumattoman poiminnan kanssa niin, että vain rajattu otos lähtötiedoista poimitaan ja näin saadaan pienemmät vertailtavat tiedostot käsiteltäväksi. Vastaavasti aikaleimaukseen perustuva poiminta voi poimia halutun otoksen tiedostoihin eri ajanjaksoina ja tiedostojen vertailuun perustuva poiminta valitsee eri ajanhetken tiedostoista tietovarastoon poimittavan otoksen (Devlin, 1997, s.190-191).

4 TIEDON YHTENÄISTÄMINEN

Operatiivisten järjestelmien tieto ei ole yhdenmukaista, sillä se saattaa tulla monista erillisistä ja toisistaan riippumattomista lähdejärjestelmistä. Tieto pitää muuttaa yhteneväksi ja yhteismitalliseksi niin, että sen käyttö tulisi olemaan ristiriidatonta. Tiedon yhtenäistämisessä tiedon laadun varmistus on erittäin tärkeä tehtävä, sillä tiedon siirto ja tallennus tietovarastoon johtaa arvaamattomiin tuloksiin, jos tieto on virheellistä tai harhaanjohtavaa. Tietovaraston sisältämä virheellinen tieto aiheuttaa yrityksessä virheratkaisuja ja voi vaikuttaa yrityksen menestymiseen ja vaarantaa jopa sen olemassaolon. Vähäisemmätkin epäluulot siitä, että tietovaraston sisältämä tieto ei olekaan oikeaa, johtaa siihen, että tietovaraston antamaan tietoon ei luoteta eikä sitä sitten myöskään käytetä.

4.1 Yhtenäinen tieto

Tiedon yhtenäistämisessä monet tietokantatuotteet avustavat joidenkin osa-alueiden rutiineissa. Tiedon muuntaminen (*transformation*) on mahdollista tehdä myös räätälöidyillä ohjelmilla. Tämä muunnosprosessi tulee tallettaa kaikkien osapuolien saataville metadatana. Sekä räätälöidyillä ohjelmilla että valmistuotteilla on hyvät ja huonot puolensa (Inmon, 1997a, s.176-178). Itse tehty liittymä antaa ohjelmoijille rajoittamattomat mahdollisuudet rakentaa liittymäohjelmisto. Toisaalta ohjelmat ovat samantyyppisiä, alituiseen muuttuvia ja niiden koodaus vaatii paljon aikaa. Ohjelmien ylläpito tulee nopeasti taakaksi. Valmiiden liittymäohjelmistojen käyttöä tukee se, että liittymä voidaan tehdä niillä nopeasti, ohjelmien ylläpito on helppoa ja nopeaa. Samalla metadata muuttuu automaattisesti, ja kustannukset pysyvät kohtuullisina. Ongelmana saattaa kuitenkin olla se, että

työvälineen oppiminen vie aikansa eikä se taivu kaikkiin tarvittaviin tehtäviin. Johtuen osaksi siitä, että tietovaraston rakentamisprojekti on iteratiivinen prosessi ja työ tehdään palasina, tietojen siirto- ja yhtenäistämisprosessi tuntuu usein yksinkertaiselta ja helpolta toteuttaa. Vaikka itse tehtävä ohjelmisto tuntuu helposti ja nopeasti rakennettavalta, se laajenee ja muuttuu monimutkaiseksi hyvin nopeasti (Inmon, 1996, s.116).

Tietovarastoon tallennettavan tietomassan yhtenäistämiseen käytettävän liittymäohjelmiston rakentaminen on vaativa ja aikaa vievä tehtävä. Tietovaraston elinkaareen nähden rakentamiskustannukset eivät kuitenkaan näyttele pääosaa, sillä ylläpidon aikaiset muutokset näyttelevät jatkossa suurempaa osaa (Inmon, 1997a, s.179). Tämä johtuu Inmonin mukaan kahdesta syystä: tietovaraston kehittäminen on iteratiivinen prosessi ja toistokierrokset vaikuttavat suoraan myös yhtenäistämisohjelmistoon. Toiseksi tietovaraston lähtötiedot tulevat operatiivisista järjestelmistä, jotka nekään eivät pysy stabiileina vaan mukailevat tuotannonohjausjärjestelmän vaatimuksia. Muutokset heijastuvat helposti myös tietovarastoliittymiin. Tiedot voidaan Inmonin mukaan (Inmon, 1997a, s.330) yhtenäistää kolmessa vaihtoehtoisessa vaiheessa. Kaikkein paras paikka käsitellä tiedot on juuri operatiivinen järjestelmä itse, jolloin tietovarastoon siirrettävä tieto on valmiiksi käsiteltyä. Toinen vaihtoehto, jota käytetään useimmin, on käsitellä tieto ennen sen siirtoa tietovarastoon. Tällöin tieto käsitellään siirto- ja latausprosessin aikana ja se onkin monesti ainut vaihtoehto. Vastuu tällöin tästä jää tietovarastoprojektin ihmisille. Kolmatta tapaa käytetään silloin, kun tietovaraston tietokantaa ei ole helppo käsitellä. Tällöin ladataan tiedot sellaisenaan sisään ja tehdään tietojen yhtenäistäminen silloin, kun niitä kysellään.

Tiedon katsotaan olevan epäpuhdasta (*dirty data*) silloin, kun tietokenttiin tallennetut tiedon arvot ovat ristiriitaisia, puuttuvat kokonaan, eivät ole luettavissa tai ovat virheellisiä. Tiedot voivat olla monella tavalla virheellistä: osoitteet voivat olla vääriä, tilauksen asiakasnumerona voi olla asiakas, jota ei ole olemassakaan tai henkilön ikä voi olla täysin suhteeton. Tällaisen epäpuhtaan tiedon käsittely tapahtuu kahdessa osassa. Ensin pitää tunnistaa, mikä tieto on epäpuhdasta ja mi-

kä ei. Se on vaativa tehtävä, sillä tietosisältö voi vaihdella paljonkin asiayhteydestä toiseen. Toiseksi tiedoille täytyy tehdä manuaalinen eheys- ja puhdistusrutiini (Mattison, 1996, s. 80).

Kun itse tieto on eheää ja yhtenäistä, sitä täytyy käsitellä sellaiseen muotoon, että se on tallennettavissa tietovarastoon. Tällöin pitää määritellä vielä kenttien koot, tyypit ja niiden järjestys tietueella. Useista eri lähteistä tuleva tietomassa täytyy yhdistää yhdeksi kokonaisuudeksi (*merge*). Pääasiallinen syy tietovaraston rakentamiselle on se, että organisaatiossa olevasta tiedosta saadaan koottu yhtenäinen näkymä. Tämä vaatii sen, että kerätylle tiedolle on pystyttävä muodostamaan yhtenäiset avainrakenteet tietovarastoon tallentamista varten. Tällainen tilanne on hyvin tavanomainen esimerkiksi silloin, kun asiakkaan tietoja kerätään eri järjestelmistä keskitettyyn tietovarastoon. Asiakas tunnistetaan usein eri tunnistetiedoilla eri järjestelmissä ja siksi tietojen puhdistus- ja yhtenäistämisvaiheessa pitäisi pystyä kokoamaan nämä kaikki tiedot saman asiakkaan alle. Tämän lisäksi kerätyistä tiedoista voidaan haluta jopa karsia jotain niin, että vain tietyn tyyppiset tai tietyn arvoiset tietueet tallennetaan (Mattison, 1996, s. 81).

4.2 Laatutavoitteet

Koska tietovarastoinnin tärkein näkökohta on kehittyä käyttäjien ja organisaation mukana, tarvitaan täsmälliset mallit tietovarastoinnin laadulle. Näiden mallien varaan voidaan sitten perustaa tietovaraston uudelleensuunnittelu ja kehitys, joka seuraa tiedon lähteiden ja käyttäjien tarpeiden muuttumista. Nämä tarpeet voidaan liittää tyypillisimpiin tietovarastoinnin laadullisiin päämääriin, joita ovat saatavuus (*accessibility*), tulkittavuus (*interpretability*), hyödyllisyys (*usefulness*), uskottavuus (*believability*) ja edellisen näkökohtien toteutumisen varmistettavuus (*validation*) (Jarke, 2000, s. 18).

Tietovarastoinnissa on kolme tahoa, jotka asettavat laatuvaatimuksia eri näkökulmista: tietovaraston ylläpitäjät, ohjelmoijat ja päätöksentekijät. Tietovaraston suunnitteluun ja ylläpitämiseen liittyvät laatutavoitteet ovat tietovaraston kuvaukseen (*schema*) ja metadatan kehitykseen (*metadata evolution*) liittyviä. Näistä kuvaukseen liittyvät tavoitteet voidaan jakaa oikeellisuuden (*correctness*), täydellisyyden (*completeness*), toistottomuuden (*minimality*), jäljitettävyyden (*traceaability*) ja kuvattavuuden (*interpretability*) tavoitteluun (Jarke, 2000, s. 139).

Laatutavoitteet tietovarasto-ohjelmistojen toteutukselle ja arvioinnille ovat toiminnallisuus (*functionality*), luotettavuus (*reliability*), käytettävyys (*usability*), tehokkuus (*software efficiency*), ylläpidettävyys (*maintainability*) ja siirrettävyys (*portability*) (Jarke, 2000, s. 139).

Kun tietovarasto on käyttäjien käytettävissä, käytön aikaiset laatutavoitteet voidaan jakaa käytettävyyteen (*accessibility*) ja hyödyllisyyteen (*usefulness*). Käytettävyydellä tarkoitetaan järjestelmän käytön mahdollisuutta muuna kuin huolto- ja ylläpitoaikoina (*system availability*) sekä latausten aiheuttamien päivitystoimenpiteiden aikana (*tranactional availability*). Siihen sisältyy myös luotettavuus (*security*), joka määrittelee valtuutukset ja käyttöoikeudet. Laadullisena tavoitteena hyödyllisyys sisältää sopivuuden (*timeliness*), riittävät vastausajat (*responsiveness*) ja ajantasaisuuden (*currency*). Lisäksi siihen luetaan kuuluvaksi tieto siitä ajasta, jona tietovaraston tieto on voimassa (*volatility*), ja tiedon tulkittavuus (*interpretability*) (Jarke, 2000, s. 140).

4.3 Tietoihin tehtävät muunnokset

Tietojen poimiminen operatiivisista järjestelmistä on vaativa tehtävä. Kuitenkin se osuus, jota ei oteta tarpeeksi huomioon tietovarastoprojekteissa on, on poi-

mittujen tietojen muunnokset. Tämä osuus vaatii suunnittelijoilta paljon laaja-alaista asiantuntemusta eri arkkitehtuurien välisistä eroista, kattavaa ymmärrystä organisaation operatiivisten järjestelmien yhteyksistä ja vastaavuuksista. Tilanne, jossa tietohallinnon atk-ammattilainen on organisaation paras operatiivisten järjestelmien integroija ja ymmärtäjä, ei ole mitenkään erikoista. Monissa tapauksissa tietovarastoprojekti onkin ensimmäinen askel organisaation vanhojen järjestelmien kartoittamiselle, tietovirtojen selvittämiselle ja dokumentoimiselle.

4.3.1 Merkkimuunnokset

Kun tieto tulee eri arkkitehtuuriympäristöistä, on siihen tehtävä vastaanottavaa ympäristöä varten merkkimuunnokset, jotta tietoa ei hukattaisi ja se näkyisi oikein. Tällainen on esimerkiksi konversio IBM-suurkonejärjestelmissä käytettävästä EBCIDIC -koodijärjestelmästä ASCII -merkistöön (Sainio, 1998, s. 41, Inmon, 1996, s.118). Muunnos voidaan monissa tapauksissa tehdä siirtotiedoston kirjoitusvaiheessa, itse tiedoston siirrossa tai vasta sitten, kun tietoa luetaan sisään vastaanottavaan järjestelmään. Suurimmat ongelmat syntyvät skandinaavisten merkkien ja numeerisissa kentissä pilkun ja etumerkin kanssa. Numeerisen kentän käsittely yksinkertaistetaan monesti selvyyden vuoksi kirjoittamalla etumerkki, kokonaisosat ja desimaalit omiksi kentiksi, jolloin virheen mahdollisuus minimoidaan.

4.3.2 Käyttöjärjestelmämuunnokset

Lähtötiedot operatiivisesta järjestelmästä sijaitsevat laajoissa sovelluksissa edelleenkin suurkonejärjestelmissä. Tällöin merkkimuunnosten lisäksi tiedot pitää pystyä siirtämään myös käyttöjärjestelmätasolla (Inmon, 1996, s.117). Koneiden välille täytyy tässä tapauksessa pystyä muodostamaan automatisoitu siirtoyhteys.

Samalla eteen tulevat myös käyttäjätunnus-, tunnistus- ja tietoturvaongelmat. Monilla ohjelmistotoimittajilla on olemassa väliohjelmistoja (*middleware*), jotka mahdollistavat esimerkiksi tietojen purkamisen (*download*) suoraan keskuskonejärjestelmän hierarkkisesta tietokantarakenteesta ennalta määrätyssä muodossa suoraan henkilökohtaisen tietokoneen tai unix-käyttöjärjestelmässä toimivaan vastaavaan tietokantaan (*upload*).

4.3.3 Tiedon sisäiset muunnokset

Kun tiedot poimitaan lähdejärjestelmästä, ne voidaan joutua hakemaan monista eri lähteistä niin, että tiedot eivät ole vielä silloin käyttökelpoisessa muodossa. Tiedosto tai tiedostot voivat sisältää tiedot kenttätasolla eri järjestyksessä kuin, mitä tietovarasto tarvitsee, tai useiden eri tiedostojen tiedot pitää yhdistää, ennen kuin ne ovat käyttökelpoiset ja yhtenäiset tietovarastoon siirtoa varten.

Inmonin (Inmon, 1997a, s.119) mukaan operatiivisten järjestelmien ja tietovaraston erilainen luonne aiheuttaa päänvaivaa. Tietovaraston tietosisällön luonne on tallettaa aikaan sidottua tietoa pitkältä ajalta. Tiedot pitää muuntaa haluttuun muotoon sillä operatiivisen järjestelmän tietojen luonne on enemmän kuluvaan ajanhetkeen ja lyhyen tähtäimen tarpeisiin liittyvää. Lisäksi tietovaraston tietosisällön tulee olla informatiivista vastakohtana operatiivisten järjestelmien yksittäisten asioiden yksityiskohtaisen tiedon antamiseen.

Pitkään elänyt operatiivinen tietojärjestelmä sisältää valtavasti sellaista tietoa, joka on tallennettu yllättävällä tavalla tietojärjestelmän tietorakenteisiin. Jotta tilaa olisi säästetty, tietokenttiin on pyritty tiivistämään tietoa antamalla tietokentän arvoille uusia merkityksiä (Devlin, 1996, s.215). Kentän arvosta voidaan saada monenlaista informaatiota, kun sen logiikan vain tietää. Hyvin yleistä on myös käyttää olemassa olevaa tietuerakennetta tallettamaan ihan jotain muuta tietoa. Erona normaaliin tietoon saattaa olla vain tietuetyypin poikkeava arvo. Kun täl-

lainen tieto halutaan siirtää tietovarastoon, täytyy kaikki useinkin dokumentoimattomat poikkeavat talletustavat tuntea ja huomioida.

4.3.4 Tiedon yhtenäistäminen

Tietosisällön tulee olla yhtenäistä ja eri lähdejärjestelmistä kerätty tieto on oltava käsitteiltään ja koodistoltaan toisiaan vastaavaa. Tietovarastoon on sovittu käytettäväksi tietty koodisto esimerkiksi osoittamaan sukupuolta. Kun syöttötiedot poimitaan monista eri lähteistä, nämä tiedot on pystyttävä yhtenäistämään niin, että lopputuloksena käytetään vain tietovarastossa sovittuja toimintatapoja.

Tietojen yhtenäistäminen on suunniteltava ja dokumentoitava hyvin, sillä se voi aiheuttaa vakavia virheitä tietovaraston sisältöön. Hyvänä esimerkkinä voi olla työnumeron muunnokset siirrettäessä tietoa toiminnanohjausjärjestelmästä tietovarastoon, mitä on kuvattu taulukossa 1.

TAULUKKO 1: Esimerkki työnumeron yhtenäistämisessä tapahtuvasta virhemahdollisuudesta siirrettäessä tietoja tietovarastoon

Operatiivisen järjestelmän työnumero	Operatiivisen järjestelmän Vaihenumero	Tietovaraston työnumero
12345	010	1234500000010
123450	010	1234500000010
1234500	010	1234500000010

Toiminnanohjausjärjestelmä sisältää työnumeron ja työn vaihenumeron erillisissä kentissä. Tietovarastoon siirrettäessä työnumero käsitetään kokonaisuudeksi, joka sisältää molemmat tietoalkiot samassa kentässä niin, että työnumero sijaitsee kentän vasemmassa laidassa ja vaihenumero aivan oikeassa reunassa. Koska vä-

liin saattaa jäädä tyhjää, se päätetään täyttää nollilla. Ajatus vaikuttaa hyvältä, mutta osoittautuu virheeksi, sillä samaan työnumeroon yhdistyy nyt useita työnumeroita taulukon 1 osoittamalla tavalla.

4.3.5 Useiden eri lähtötietojen yhdistäminen

Tietovarastoon siirrettävä tieto saattaa löytyä hyvin monesta operatiivisen järjestelmän lähtötiedoista. Koska operatiiviset järjestelmät on rakennettu eri aikoina ja useinkin toimivat täysin erillään toisistaan, niistä saatavat tiedot ovat ristiriitaisia keskenään. Tällöin tietovarastoon siirrettäessä on päätettävä, mistä haluttu tieto pääasiassa haetaan. Operatiivisessa järjestelmässä kaikkia tietoja ei välttämättä ole edes tallennettu, mikä johtaa siihen, että siirrettäessä niitä tietovarastoon, niille on pystyttävä antamaan oletusarvoja tai hakemaan puuttuvat tiedot jonkun toisen järjestelmän tiedoista. Inmonin mukaan ehkä kaikkien vaikein tehtävä onkin ymmärtää vanhojen operatiivisten järjestelmien logiikka niin, että poimitut lähtötiedot voitaisiin käyttää luotettavasti tietovaraston syöttötietoina (Inmon, 1996, s. 117, Inmon, 1997a, s.119).

4.3.6 Summaus ja yhdistely

Yksi tietovarastoinnin voimakkaimmista ominaisuuksista on kyky summata tai yhdistellä tietoja. Operatiivisen järjestelmän tiedot on monesti tallennettu täysin tapahtumakohtaisina. Tällaisina niitä ei kannata tallentaa tietovarastoon vaan ne voidaan tässä vaiheessa koota suuremmiksi kokonaisuuksiksi. Tallennettavan tiedon rakeisuutta voidaan muuttaa tietovaraston vaatimusten mukaisiksi, jotta eri lähteistä saatava tieto olisi yhteismitallista. Useimmat operatiivisen järjestelmän tiedot sisältävät tapahtumahetken aikaulottuvuuden. Jos tällainen kuitenkin

puuttuu, se voidaan tässä yhteydessä antaa, sillä tietovarastossa useimmat tiedot vaativat kiinnityksen aikaulottuvuuteen. Vanhemmissa järjestelmissä ajan tallentaminen vaihtelee paljon. Aika -käsitteelle on löydyttävä yhtenäinen muoto, jossa otetaan huomioon myös vuosituhannen vaihtuminen ja mahdollisesti vaadittava kellonaika. Luettavan tiedon muoto on eri lähdejärjestelmissä erilainen (Inmon, 1996, s.118). Tiedot voidaan joutua purkamaan erillisiksi elementeiksi muuttuvan tai kiinteän pituista tietueista, joissa ne voivat olla erotettuna toisistaan sovitulla merkillä. Tietoalkioiden määrä voi vaihdella ja ne on voitu esittää myös taulukkorakenteena.

4.4 Virheellisen tiedon hallinta

Suurin ryhmä todellisen elämän ongelmatapauksista johtuu liiketoiminnan sääntöjen rikkomisesta. Esimerkiksi työsuhteen päättäneen henkilön tiedoissa saattaa näkyä toistuvia maksusuorituksia. Tilanne voi olla oikea tai virheellinen, mutta sen oikeellisuuden voi tietää vain asiayhteyden tunteva vastuuhenkilö. Ohjelmatoimittajat lupaavat, että konversio-ohjelmilla voidaan yhtenäistää ja korjata 95% tiedoista. Jotta päästäisiin täydellisempään lopputulokseen, vaaditaan seuraavat tekijät. Tässä prosessissa on oltava tietohallinnon henkilöiden lisäksi liiketoiminnan tunteva asiantuntija, joka osaa käyttää ja määritellä liiketoiminnan säännöt, joita yhtenäistämisessä käytetään. Uusia virheitä ei saa laskea läpi vain luottamalla siihen, ne toimivat testaamattomina. Lisäksi on huomattava, että yhden liiketoiminnan säännön muuttaminen voi aiheuttaa virhetilanteen johtamalla tiedot ristiriitaan jonkun muun liiketoiminnan säännön kanssa. Jotta tietojen yhtenäistäminen ei kestäisi liian kauan, on siinä käytettäviin välineisiin panostettava ja pyrittävä nopeaan läpimenoon. Tiedon latauksen tulisi sallia kaikki tiedon muodot peräkkäistiedostosta aina relaatiotietokantarakenteisiin. On tärkeää säilyttää lähdejärjestelmän koskemattomuus, sillä se helpottaa mahdollisen virheellisen tiedon syyn ja alkuperän etsintää (DmDirect, Arkady Maydanchik, 1999).

Virheellinen tieto on pystyttävä suodattamaan pois, ja eliminoimaan duplikaatit. Inmonin (Inmon, 1996, s.119) mukaan vaikeinta virheellisen tiedon hallinnassa onkin se, että tietojen tulkintavirheiden estämiseksi tietovarastosuunnittelijan olisi tunnettava operatiivisen järjestelmän toimintalogiikka perinpohjin.

Virheellinen tieto on lähes aina väärin myös operatiivisessa järjestelmässä. Vikaa voi olla myös siinä ohjelmassa, joka kirjoittaa tiedot tietovaraston suuntaan esimerkiksi peräkkäistiedostoon. Myös tietovaraston tietoihin tehtävät muutostoimenpiteet voivat aikaansaada virheen. Devlin (Devlin, 1997, s.217) huomauttaakin, että virheellisen tiedon korjaamista on vältettävä. Virheet joutuu korjaamaan lähes aina operatiivisten järjestelmien vastuuhenkilö. Devlinin mukaan virheitä on kolmenlaisia. Tietojen tietosisältö (*content error*) voi olla virheellinen. Tähän paras keino on tarkentaa tietojen syöttöjärjestelmien tarkistusrutiineja niin, ettei se salli virheellisen tiedon tallennusta. Virheet tietojen käytössä (*usage error*) kertovat vakavista puutteista operatiivisissa järjestelmissä. Tällaisia virheitä ei kannata edes yrittää korjata tässä vaiheessa, vaikka joskus se tuntuisikin helpommalta ratkaisulta kuin yrittää saada korjattua virheen aikaansaavaa järjestelmää. Kolmas virhetyyppi on suunnitteluvirhe operatiivisessa järjestelmässä (*design error*). Sen ei tarvitse välttämättä olla vakava virhekään, mutta se osoittaa, että ongelmia on itse operatiivisen järjestelmän sisäisissä rutiineissakin.

4.5 Ristiviittausten tarkastaminen

Latausta odottavan tiedon tietosisällön on käytettävien koodien osalta vastattava sillä hetkellä käytettäviä koodeja. Tällaisten ristiviittausten tarkastaminen latausvaiheessa hidastaa latausprosessia ja aiheuttaa tietojen eheysongelmia. Tietokannanhallintajärjestelmän itsensä tekemät, lähinnä indeksien ja muiden kenttäkohtaisten viittausten tarkastaminen taulun muihin kenttiin tai toisten taulujen kenttiä

vasten, ovat ongelma. Latausten nopeuttamiseksi ja nopeiden latausmenetelmien käyttämiseksi (esimerkiksi Oraclen sql-loader), tällaiset rajoitukset joudutaan jättämään huomiotta latauksen aikana. Kun tiedot on ladattu relaatiotauluun ja indeksit ja viittaustiedot kytketään päälle, tiedot eivät välttämättä ole eheät. Yksikäsitteiset (*unique index*) indeksit eivät olekaan kunnossa, eikä esimerkiksi tilauksen tilausrivillä olevaa tuotenumeroa löydykään tuotetaulusta. Kun tietovarastoon kerätään tietoa monista eri lähdejärjestelmistä, tietojen latauksen järjestys ja ajoittaminen on tärkeää edellä mainittujen ongelmien välttämiseksi. Siltikin ristiviittausten ja eheyden hallinta pitää olla suunniteltua. Esimerkiksi tietoliikennekatkosten takia jonkun lähdejärjestelmän tieto voi puuttua tietovarastosta useita päiviä eikä se saa estää ainakaan kaikkien muiden tietojen latausta. Jos tällainen tilanne sallitaan, se saattaa aiheuttaa virheitä yksittäisissä tietoalkioissa ja myös summatuissa tiedoissa.

5 TIEDON LATAUS TIETOVARASTOON

Arkkitehtuurisena valintana tiedon lataus tietovarastoon voi tapahtua yksi-, kaksi- tai kolmitasoinen latausrakenteen mukaan. Näillä kullakin on omat etunsa ja soveltuvat käytettäviksi määrätyissä tilanteissa.

Useimmat tietovarasto-ohjelmistotoimittajat suosittelevat välivarasto –tiedostojen tai –taulujen käyttöä ennen kuin tieto varsinaisesti tallennetaan tietovarastoon. Tässä ns. kokoamisalueella (*staging-area)* tieto on jo loppuun asti käsiteltyä ja ainut toimenpide on tallettaa se tietovarastoon. Kimballin (1996, s.215) mukaan tietovarastoon ladattava tieto on lähes aina otos operatiivisesta järjestelmästä. Tiedon saanti operatiivisesta järjestelmästä on usein työläämpää kuin uskotaankaan: tämän osuuden viemä aika arvioidaankin kaksi tai kolme kertaa todellisuutta lyhyemmäksi. Kimball suosittelee tietovarastosovelluksen arkkitehtuuriratkaisuksi peilattuja levyjä kahdesta syystä. Päiväaikaan tosiaikakäytössä käyttöjärjestelmä pystyy jakamaan kyselykäytön kuormaa tasaisemmin molemmille levyille ja peilattu levy varmistaa samalla katkottoman käytön, jos toinen levy vioittuu. Yöaikaan, kun tietovarastoon ladataan tietoa, kytketään toinen levy pois latauskäytöstä. Silloin se sisältää edellisen päivän varmistuksen ja pystyy toimimaan kyselykäyttötarkoituksessa myös latausikkunan aikana. Jos lataus epäonnistuu ja joudutaan palaamaan edellisen päivän varmistukseen, se tapahtuu kätevästi palauttamalla toisen levyn tilanne. Jos taas latausprosessi onnistuu, otetaan molemmat pelilevyt käyttöön, jolloin tietojen päivitys toiseen peilattuun levyyn tapahtuu käyttöjärjestelmän toimesta hyvin nopeasti.

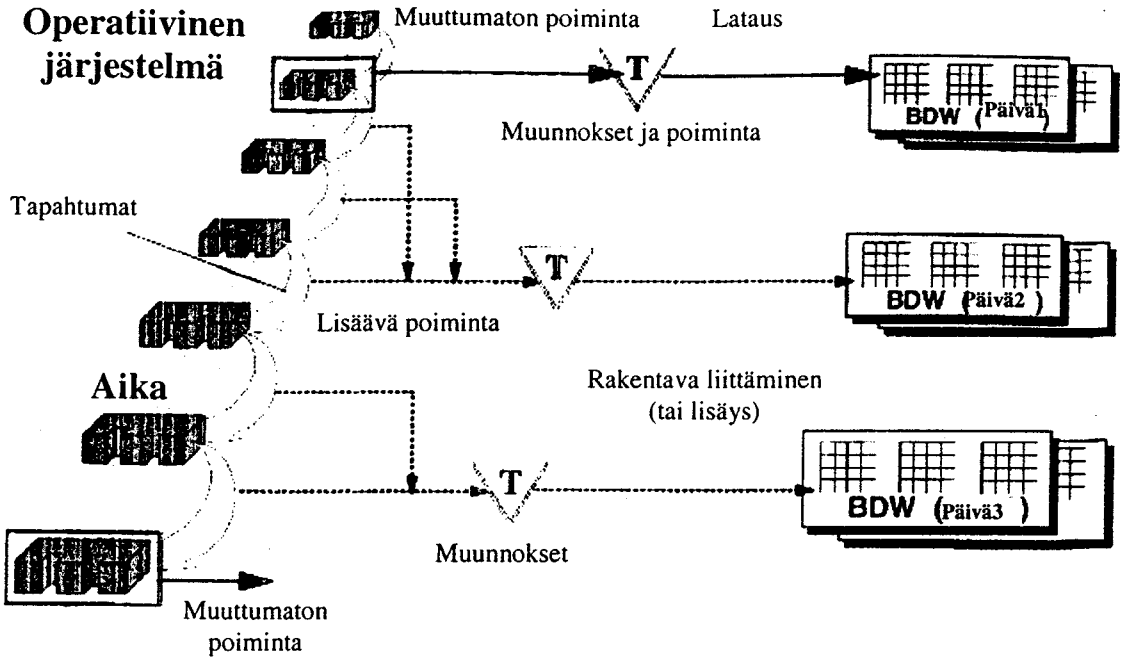
Tietovaraston latausprosessi voidaan jakaa karkeasti alkulataukseen ja jaksottaisiin muutosten latauksiin. Alkulataus on tietovaraston pystytyksen yhteydessä tehtävä ensimmäinen tietojen tallennuskerta. Myöhemmin tietovaraston ylläpidon

yhteydessä tehtävät latauskerrat ovat normaaleja operatiivisen järjestelmän muuttuneiden tietojen tallentamista tietovarastoon.

Muutostietojen säännöllisissä latauksissa on periaatteena tuoda operatiivisten tietojärjestelmien muuttunutta tietoa tietovarastoon, missä se saa sille kuuluvan ajallisen ulottuvuuden. Varsinkin dimensiotaulujen muuttuneen tiedon havaitseminen ja hallinta on tärkeää, jotta haettavat tiedot vastaisivat aina kutakin ajallista ulottuvuutta. Tähän löytyy erilaisia tekniikoita, joilla tietojen muutosten aiheuttamat ongelmat hakukriteereissä ratkaistaan (surrogaattiavaimet, tietojen korvaaminen uudella tiedolla ja uuden kentän lisääminen). Inmonin mukaan jatkuvasti muuttuvan tiedon havaitseminen, poiminta ja lataaminen tietovarastoon ovat tietovarastoarkkitehtuurille suurin haaste (Inmon,1996, s.77). Muutosten poiminta, tehtävät muunnokset ja lataus tietovarastoon eivät ole helppo tehtävä.

Tietojen tallennus tietovarastoon voidaan jakaa neljään erilaiseen tyyppiin (Devlin, 1997, s.196): lataus (*apply*), lisäys (*append*), tuhoava liittäminen (*destructive merge*) ja rakentava liittäminen (*constructive merge*). Lataus ja lisäys voidaan katsoa kuuluvaksi tietovaraston luomiseen eli alkulataukseen. Tietovaraston normaaleihin tietojen jaksottaisiin latauksiin kuuluvat selkeästi tuhoava ja rakentava liittäminen. Tiedon tallennuksen tyypit on kuvattu kuviossa 14.

Kokonaisuutena tietovaraston rakentamista, alkaen alkulatauksesta, jatkuen käytön aikaisiin jaksottaisiin latauksiin, voidaan kuvata seuraavalla kuvalla. Siinä on esitetty myös tietojen yhtenäistäminen ja siirto tietovarastoon. Viimeisenä vaiheena kuvattu muuttumaton poiminta merkitsee sitä, että aika-ajoin olisi hyvä ladata operatiivisten järjestelmien alkutilanne täysin uudestaan alkulatauksen tyyppisenä latauksena (Devill, 1996, s.218).

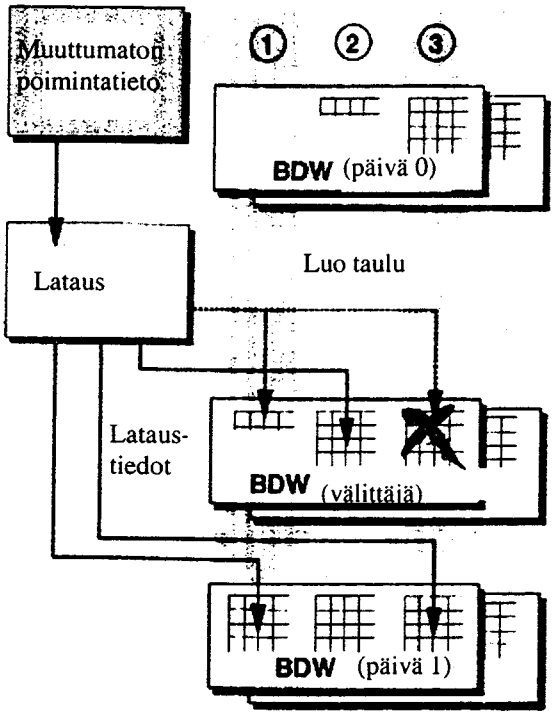


Kuvio 14. Tietojen poiminta, puhdistus, siirto ja lataus tietovarastoon (Devlin, 1997, s. 218)

5.1 Alkulataus

Alkulataus poikkeaa normaalin lataussyklin mukaisista latauksista, sillä silloin tieto haetaan operatiivisen järjestelmän historiatiedoista ja varmuuskopioilta. Ladattavan tiedon käsittely saattaa vaatia omat, tätä toimenpidettä varten räätälöidyt, ohjelmansa. Inmon jakaa alkulatauksen vielä kahteen osaan (Inmon,1996, s.77). Alkulataus voidaan tehdä varmuuskopioilta, jolloin kyseessä on vanhan historiatiedon lataus. Tätä alkulatauksen osuutta ei Inmonin mukaan kuitenkaan usein tehdä, koska se on työlästä eikä siitä saada kovinkaan paljon lisäarvoa tietovaraston sisältöön. Alkulataus tehdään lähes aina pelkästään olemassa olevasta operatiivisten järjestelmien tiedoista kirjoittamalla tiedot peräkkäistiedostoihin ja siirtämällä ne tietovarastoon. Tällä tavalla aiheutetaan mahdollisimman vähän häiriötä käytössä oleviin järjestelmiin.

Devlinin mukaan alkulataus koostuu useista erillisistä toimenpiteistä, jotka johtavat tietokannan ja sen taulujen suunnittelusta fyysisen tietokannan perustamiseen. Tämä tapahtuu ensiksi case-välineellä, joka muodostaa tietokannan luontilauseet muodostettavalle tietokannalle. Toisena vaiheena replikointiväline luo taulujen luontilauseet. Toinen vaihtoehto on luoda taulut suoraan tietojen tallennuksen yhteydessä 'CREATE TABLE AS SELECT' -lauseella. Heti tämän jälkeen luodaan tarvittavat indeksit (Devlin, 1997, s.197).



Kuvio 15. Alkulatauksen tallennustapa: lataus (Devlin, 1997, s. 198)

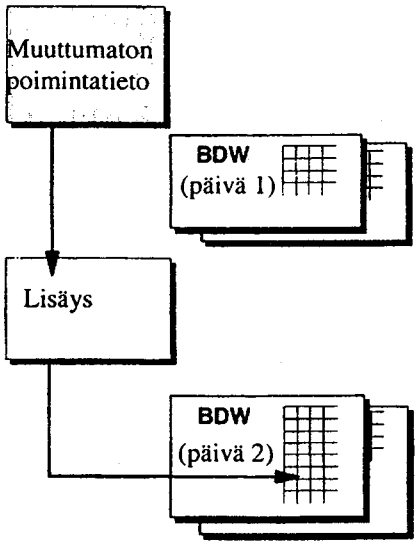
Kuviossa 15 kuvataan tietojen tallennustavoista lataus (*load*) ja lisäys (*append*). Ne ovat alkulatauksessa käyttökelpoiset tavat siirtää tietoa tietovarastoon. Lataus on yksikertaisin ja laajin tapa. Se tallettaa siirrettävän tietomassan niin, että

1. jos taulua ei löydy, niin se luodaan ja tiedot ladataan siihen
2. jos taulu jo löytyy, tiedot ladataan siihen
3. jos kohdetaulussa on jo tietoja, ladattavat tiedot korvaavat olemassa olevat tiedot täydellisesti. Tämä tapahtuu joko poistamalla koko taulu (drop table) ja

luomalla se uudelleen tai poistamalla olemassa olevat tiedot ja kirjoittamalla tietueet uudelleen.

Lataus käyttää syöttötietonaan muuttumattoman poiminnan tietoja (vrt. Devlin, 1997, s.182) asettaen tallennettavat tietueet alkuarvoja vastaavaan tilaan. Teknisesti syöttötietoina voisi olla myös lisäävän poiminnan aikaa saama syöttötiedosto, mutta siinä tapauksessa tietueet voisivat saada virheelliset alkuarvot (päivitetty / poistettu) (Devlin, 1997, s.198).

Toinen muuttumatonta syöttötietoa käyttävä tietojen tallennustapa on lisäys (*append*), joka on kuvattu kuviossa 16. Se voidaankin nähdä latauksen muunnoksena



Kuvio 16. Alkulatauksen tallennustapa: lisäys (Devlin, 1997, s. 199)

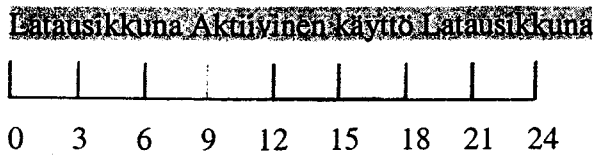
siinä tapauksessa, jossa ladattava tieto on jo tietovarastossa. Erona lataus - tallennustapaan on se, että tietovaraston kohdetauluja ei tyhjennetä vaan tiedot ladataan joko huomioimatta aikaisempia tietoja tai hylkäämällä ne tiedot, jotka jo ovat kohdetauluissa. Lisäyksen käyttämisessä tulee olla tarkkana, sillä saman aineiston uudelleen lisäys voi aiheuttaa tietovaraston eheyden rikkoutumisen, jos samalla avaimella kirjoitetaan tietueita moneen kertaan. Normaalisti tietokannan

yksikäsitteiset avaimet estävät duplikaattien syntymisen, mutta jos taulun indeksit poistetaan ennen latausta ja muodostetaan uudelleen latauksen jälkeen suoritustehon parantamiseksi, duplikaattitietueiden synty on mahdollista. Myös samaan tauluun kirjoitettavien, mutta eri operatiivisista järjestelmistä saatavien tietojen päällekkäiset avainrakenteet aiheuttavat monimutkaisia tilanteita. Esimerkiksi sama asiakas saattaa olla koodattuna erilaisella avainrakenteella eri lähdejärjestelmissä. Toisaalta samalla asiakasnumerolla saattaa löytyä kaksi kokonaan eri asiakasta, kun tiedot poimitaan eri paikoista. Nämä tilanteet eivät aiheuta ongelmia, kun ne otetaan huomioon ja toimitaan konfliktitilanteessa suunnitellulla tavalla (Devlin, 1997, s.199-201).

5.2 Jaksottaiset lataukset

Säännöllisesti tehtävät lataukset lähinnä operatiivisista järjestelmistä tapahtuvat normaalisti enintään kerran vuorokaudessa. Inmonin (Inmon, 1996, s.114) mukaan 24 tunnin jakso on suositeltava jaksotus, sillä mitä useammin tietoja päivitetään tietovarastoon, sitä kalliimmaksi ja monimutkaisemmaksi järjestelmä tulee. Vuorokauden jaksotus voidaan tehdä vielä perinteisillä teknologialla. Tätä tärkeämpänä syynä Inmon pitää kuitenkin sitä, että harvempi päivitystiheys vähentää tarvetta tehdä operatiivisen järjestelmän kaltaisia toimintoja tietovarastossa, mutta myös vastaavasti tietovaraston kaltaisia toimintoja operatiivisessa järjestelmässä. Vuorokauden syklin aikana operatiivisen järjestelmän tiedot ehtivät vakiintua ennen kuin ne siirretään tietovarastoon.

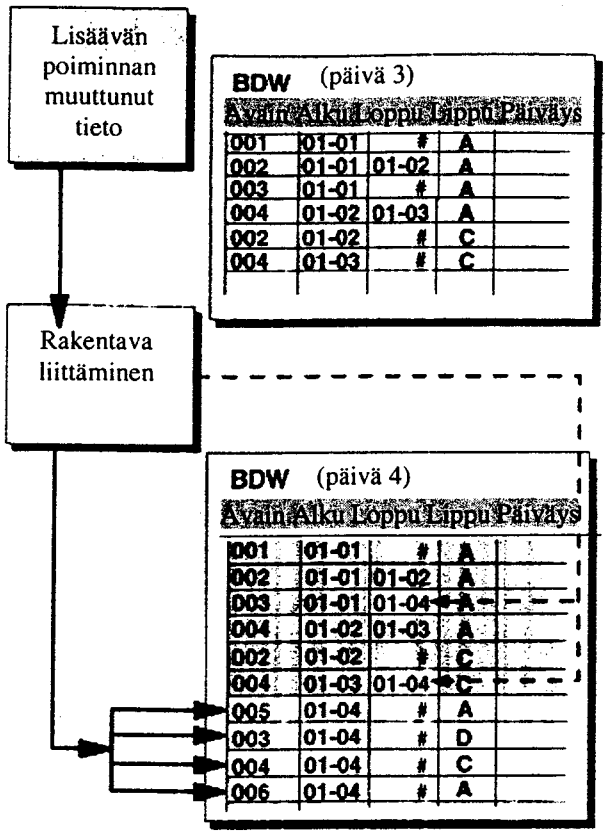
Latausikkuna (*load window*) on se aika, joka tarvitaan tietojen poimimiseen, yhtenäistämiseen, lataamiseen, jälkikäsittelyyn ja saattamiseen tietovaraston käyttäjien saataville (Data Warehousing fundamentals Volume 1-3, 1996, s. 9-14) (kuvio 17).



Kuvio 17. Latausikkuna (Data Warehousing fundamentals Volume 1-3, 1996, s. 9-14).

Tänä aikana tietovarasto ei ole kyselykäytössä. Tiedon latausten tulee olla tehokkaita, sillä huoltoikkunan aikana tehdään myös varmuuskopioiden ottamisen lisäksi myös tietokannan huoltoon liittyviä tehtäviä. Latausikkunan muita tehtäviä ovat latausajojen onnistumisen analysointi, indeksien ja summataulujen luominen, metadatan päivittäminen, tietovaraston varmuuskopioiden ottaminen, näkymien (*view*) luominen mahdollisia käyttäjien tarpeita varten sekä tietovaraston ja sen summataulujen käytön salliminen (Data Warehousing fundamentals Volume 1-3, 1996, s. 9-21).

Tietovaraston ylläpidon aikaisessa jaksottaisessa latauksessa tärkeä piirre on se, että olemassa olevia tietoja ei korvata vaan niitä pääasiassa vain kirjoitetaan lisää. Tällöin kirjoitettavien tietueiden avainrakenne on oltava sellainen, että päällekirjoitus ei ole mahdollinen. Rakentava liittäminen (*constructive merge*), mikä on kuvattu kuviossa 18, ottaa huomioon tämän, sillä siinä käsiteltävä avain koostuu sekä perusavaimesta (*business key*) ja siihen liittyvästä aikaleimoista (*timestamp*). Aikaleimat koostuvat sekä alku- että loppuajasta. Rakentava liittäminen koostuu kahdesta toiminnosta. Ensimmäinen osuus kirjoittaa tiedot tietovarastoon. Vaikka perusavain olisikin sama, aikaleima pitää tietueet erillään tietokantataulussa. Tietueella tulee myös operatiivisen järjestelmän toimenpidettä osoittava tilakoodi (muutos, lisäys tai poisto). Toisessa vaiheessa käydään läpi samalla perusavaimella olevat tietueet. Vanhimman alkuaikaleima jää ennalleen, mutta tietueen loppuaikaleima päivitetään vanhimman tietueen aikaleimalla; myös tilakoodi päivitetään. Tällä tavalla tietuella on lopuksi perusavain, alku- ja loppuaikaleimat ja viimeisin operatiivisen järjestelmän tilatieto (Devlin, 1997, s.201-203). Devlinin mukaan edellä kuvattu rakentavan liittämisen toimintaperiaate on par-



Kuvio 18. Jaksottaisen latauksen tallennustapa: rakentava liittäminen (Devlin, 1997, s. 202)

haimmillaan silloin, kun muuttuneita tietueita on suhteellisen vähän. Tämän vuoksi voikin olla tarpeen siirtää muuttuneet tietueet useammin kuin kerran vuorokaudessa, jotta tietuelukumäärä pysyisi mahdollisimman pienenä.

5.3 Tietovaraston päivittäminen

Tietovaraston tietojen päivittämisen (*refreshment*) katsotaan normaalisti sisältävän latausprosessin (*loading process*) ja muuttuneiden tietojen päivittämisen tietovarastoon (*view maintenance*). Tietojen päivittäminen onkin tärkeä prosessi, sillä se määrää poimittujen ja summattujen tietojen tehokkaan käytettävyyden. Lisäksi se aika, mikä menee muuttuneen tiedon poimimiseen ja toimittamiseen tietovarastoon, määrää tietovaraston tietojen käytettävyyden laadun. Bouzeghoub, Fabret ja Matulovic-Broqué (1999) ovat sitä mieltä, että tietojen hankinta ei ole pel-

kästään muuttuneiden tietojen päivitysongelma tai sekava latausohjelmien kokoelma, vaan kyseessä on monimutkaisempi ongelmakenttä. Heidän mukaansa se on työnkulku (*workflow*), jonka toiminnot riippuvat käytössä olevista tuotteista tiedon hankintaprosessissa, eikä sitä saa pitää pelkästään tietojen latausprosessina. Tiedon hankinta on heidän mielestä tiedon poimintaa, puhdistusta, yhtenäistämistä ja käyttöönottoa. Näiden tapahtumat ajoittaminen (*triggering event*) riippuu sovellusalueesta ja tietojen päivittämisen vaatimustasosta.

Tietovirta, joka kuvaa latausprosessia, antaa pohjan tietojen päivittämiselle, mutta niissä työvirta on erilainen kuin latausprosessilla. Päivitysprosessi on dynaaminen ja kehittyy käyttäjän tarpeiden ja lähdejärjestelmien muuttumisen myötä, kun taas tietovaraston alkulataus on staattinen ja on määritelty käyttäjien ja lähdejärjestelmien nykyvaatimusten mukaisesti. Päivitys saattaa olla täysin tahdistettu tiedon hankinnan osatehtävien kanssa. Toiseksi tiedon poiminta lähdejärjestelmistä saattaa olla rinnakkaista niin, että jokaisella tiedon lähteellä on oma huoltoikkunansa. Tahdistus tietovarastoon tapahtuu sitten vasta integrointivaiheessa. Lisäksi erona on lähdejärjestelmien käytettävyys. Kun latausvaihe vaatii paljon lähdejärjestelmien aikaa, niin päivityksen ei katsota rasittavan operatiivisia järjestelmiä tietojen purkuvaiheessa. Latausprosessin katsotaan kuuluvan tietovaraston rakentamiseen, ja sen aiheuttama käyttökatko hyväksytään. Päivitykselle asetetaan suurempia vaatimuksia käytettävyyden kannalta; silloin tietovaraston ja lähdejärjestelmien ei katsota edellyttävän käyttökatkoja.

Työvirta päivitysprosessissa on samankaltainen kuin latausprosessissakin. Erona näissä on ainoastaan se, että päivitys käsittelee oleellisesti pienempää tietomassaa kuin lataus. Työvirran katsotaan koostuvan valmistelu- (*preparation*), koonti- (*integration*), summaus- (*aggregation*) ja käyttöönottovaiheista (*customzation*). Valmisteluvaiheessa vain muuttunut tieto poimitaan lähdejärjestelmistä niiltä osin, mitä muutoksia on tapahtunut viimeisimmän poiminnan jälkeen. Tiedot yhtenäistetään ennen kuin ne siirtyvät koontivaiheeseen. Koontivaiheessa eri lähteistä tulleet tiedot yhdistetään. Summausvaiheessa lasketaan uudelleen halutut sum-

maustasot ja yhdistellään tietoja. Käyttöönottovaihe siirtää käsitellyn tiedon käyttäjien käytettäväksi.

Työvirtaa ohjataan ja koordinoidaan erilaisilla menetelmillä, jotka voivat perustua prosessien antamiin hyväksymis- tai sähköposti-ilmoituksiin, väliaikaisiin tapahtumiin tai muihin ajoitettuihin tapahtumiin. Päivitystoimenpiteitä koordinoidaan väliaikaisilla tapahtumilla (*temporal event*), päättymistä osoittavilla tapahtumilla (*termination event*) tai käyttäjän määrittelemillä tapahtumilla (*user-defined event*). Työvirta koostuu lähes aina toisiaan seuraavista tapahtumista, jotka on suoritettava peräkkäin ja vasta sitten kun edellinen tapahtuma on saatu loppuun. Toisaalta tapahtumaa ei suoriteta ollenkaan, jos tarvittava impulssi puuttuu. Näitä ohjaa kaksi työvirta-agenttityyppiä: inhimilliset ja tietokoneavusteiset agentit. Inhimillisiin agentteihin luetaan vaatimukset, rajoitteet ja strategiat, joita tietokannanhoitajat ja tietojärjestelmäasiantuntijat määrittelevät. Tietokoneavusteisiin agentteihin taas kuuluvat lähdejärjestelmiä ja tietovarastoa valvovat järjestelmät.

6 YHTEENVETO

Tietovarasto on organisaation eri lähdejärjestelmistä kerättyä yhdenmukaistettua tietoa, joka kuvaa avainprosesseihin liittyvää tietämystä. Lähdejärjestelmät voivat olla sekä sisäisiä että ulkoisia. Tietovarasto sisältää tietoa pitkältä ajalta. Periaatteena onkin ladata tietovarastoon vain uutta tietoa; vanhoja tietoja ei päivitetä. Näin ollen tietovarasto sisältää valtavan määrän tietoa, mikä aiheuttaa resurssiongelmia. Resurssiongelmat liittyvät sekä tietojen alati kasvavaan tilatarpeeseen, hallinnointiin, varmuuskopioihin että tietojen hakuaikoihin. Tiedon määrä on valtava eikä ongelmana usein olekaan sen saanti vaan valinnan vaikeus: samaa tietoa löytyy useasta eri lähdejärjestelmistä eri asiayhteyksiin kytkettyinä ja eri tavoilla koodattuna. Lisäksi kerättävän tiedon aikaulottuvuus vaihtelee ja aiheuttaa yhteensopivuusongelmia tietovaraston tallennuksen yhteydessä.

Tiedot tallennetaan tietovarastossa faktatauluun, joita voi olla yksi tai useampi. Faktataulu sisältää kaiken tietovaraston summattavissa olevan tiedon sekä viiteavaimet dimensiotauluihin. Dimensiotaulut eivät sisällä varsinaisesti tietoa vaan ne ovat lähinnä tietojen lähestymisen näkökulmia. Faktataulun tiedot haetaan dimensiotaulujen perusteella. Mitä useamman dimensiotaulun tiedon perusteella haku tapahtuu, sitä pienempi on saatava tietuelukumäärä. Tietovaraston vanhimpia tietoja voidaan tiivistää, jolloin yksityiskohtaiset tiedot joko summataan, siirretään toiselle tietovälineelle tai jopa poistetaan. Tietovarastosta haettava tieto on usein miten jollakin tavalla yksittäisistä tietuista summattua tietoa. Tämän vuoksi tietovarastossa on erityisiä summatauluja, joihin on jo etukäteen laskettu usein haettavaa summatietoa, mikä nopeuttaa oleellisesti tietojen hakua. Tietovaraston taulurakenteiden suunnittelu on vaativa tehtävä, joka vaikuttaa oleellisesti tietovaraston käyttöön ja kehittämiseen. Kun ensimmäistä tietovarastosovellusta rakennetaan, kannattaa aloittaa yhdestä helposta osa-alueesta, jolla kerätään kokemusta. Tietorakenteita suunnitellessa kannattaa tutustua valmiisiin ratkaisuihin ja huomioida niiden toteutuksessa havaitut ongelmat, jotta ne vältet-

täisiin. Ratkaisut on oltava sellaisia, joiden kanssa on helppo elää vuosia, sillä tietovarasto muuttuu jatkuvasti.

Tietovarastointi käsittää kolme perustehtävää: tiedon hankinta, tallennus tietovarastoon ja sen esille saanti käyttäjien käytettäväksi. Tiedon hankinta tietovarastoon koostuu tietojen poiminnasta lähdejärjestelmistä, yhtenäistämisestä sekä siirtämisestä ja lataamisesta tietovaraston tietokantaan. Tiedon hankinta on normaalisti yöllä kerran vuorokaudessa tapahtuvaa toimintaa, jolloin tietovarasto ei ole kyselykäytössä. Se voi ainakin joiltakin osiltaan olla myös lähes tosiaikaista tietojen replikointia operatiivisista tietojärjestelmistä tietovarastoon. Tiedon hankinnan suurin ongelma on kerätä tietoa organisaation hyvinkin erillisinä kehitetyistä tietojärjestelmistä, yhdenmukaistaa ja puhdistaa se. Tietojen oikeellisuuden ja yhteismitallisuuden varmistaminen vaatii suuria ponnistuksia ja yhteistyötä. Tämä tehtävä on käytännössä monimutkainen ja vaikea toteuttaa. Rutiini on pystyttävä tekemään hyvin yksinkertaiseksi ja varmaksi sekä tietojen poiminnan, siirtämisen että lataamisen osalta. Jotta tähän päästäisiin, on toimintaketju dokumentoitava hyvin, varmistettava virhetilanteiden käsittely ja havaitseminen ja tehtävä uusinta-ajot mahdollisimman yksinkertaisiksi. Käytäntö opettaa hyvin nopeasti, että toimintaketjun on oltava hyvin selkeä, sillä virheitä tapahtuu joka tapauksessa. Niitä aiheuttavat poikkeustilanteet kuten koneiden alasajot, huoltokatkot, vuodenvaihteen tuomat ongelmat, levytila- tai tietokantatilojen loppumiset, ohjelmavirheet ja muut inhimilliset tekijät.

Tietojen yhtenäistäminen voidaan tehdä tietokantatoimittajien valmiilla työkaluilla tai räätälöidyillä ohjelmilla. Yhtenäistäminen vaatii muutoksia tietosisältöön, jolloin tietoihin voidaan joutua tekemään merkkimuutoksia, käyttöjärjestelmien vaatimia muutoksia, tiedon sisäisiä muunnoksia, tietojen yhtenäistämistä, useiden eri lähtötietojen yhdistämistä, summausta ja yhdistelyä. Oleellisia tehtäviä ovat myös virheellisten tietojen hallittu käsittely ja ristiviittausten tarkastaminen. Valmiit tietokantatuotteet vaativat usein hyvin määrämuotoisen ja standardin lähtötiedoston ja selkeät muutossäännöt. Lähtötietojen suuren vaihtelevuuden vuoksi joudutaan usein päätymään räätälöityjen ohjelmien käyttöön. Se on työ-

lästä ja vaativaa, sillä latausrutiinin pitää olla yksinkertaista ja suoraviivaista eikä saa muuttua epävarmaksi ja dokumentoimattomaksi puutteellisten tietojen täydentelyksi. Tietojen lähde ja tehtävät muunnokset ja tarkastukset on oltava helposti selvitettävissä.

Yrityksissä, joiden tietojärjestelmät ovat kehittyneet vuosien varrella yritysostojen, useiden erilaisten organisaatiomuutosten ja itsenäisten organisaation osien kehittämistyön tuloksena, yhtenä vaikeimpana ongelmana on koko organisaation kattavan yhtenäisen tiedon hallinta. Vaikka tietojärjestelmät ovatkin toimivia ja niiden yhteydet esimerkiksi konsernin sisällä toimivat, tulee jatkuvasti esiin tarve pystyä keräämään koko organisaatiota koskevaa yhtenäistä tietoa laaja-alaisesti. Asia tulee esille joka kerta, kun tietoa siirretään organisaation sisällä yksiköstä toiseen. Tilanne näyttää nopeasti katsoen toivottomalta ja mahdottomalta ratkaista. Sitä se itse asiassa onkin ilman koko organisaation kattavaa tietojärjestelmää. Vaikka yksittäinen organisaation osa rakentaisikin tietovaraston, joka kerää tietoa ja yhtenäistää sen, tuntuu tehtävä toivottomalta joskus mahdottomienkin muunnossääntöjen ja konversioiden vuoksi. Tietovaraston ylläpitäminen on oltava rutiininomaista ja selväpiirteistä, vaikka siinä varaudutaankin yhtenäistämään ja tekemään tieto yhteismitalliseksi.

Tietomäärät on valtavia ja tietojen tallennusaikavaatimukset ovat useita vuosia. Tietomassan hallinta vaatii tietovaraston kaltaisen toimintamallin käyttämistä, jotta järjestelmä olisi käyttökelpoinen. Monissa tapauksissa tietovaraston merkitys on enemmänkin tietojen hakuaikojen nopeuden varmistaminen, tietokannan suuren tietomassan tiivistämisen ja vanhan tiedon poistamisen varmistaminen tehokkaalla tavalla. Tiedon hakeminen eri näkökulmista ei välttämättä ole kaikkein tärkein tavoite.

Tutkielma on selvittänyt tietojen poimintaa, yhtenäistämistä ja lataamista tietovarastoon. Monet organisaatiot toimivat nykyisin maan- tai maailmanlaajuisesti. Tietojärjestelmät sisältävät epäyhtenäistä tietoa koska keskitetty tietovarasto puuttuu. Koska tietovarastot keräävät tietoa monista tietojärjestelmistä, tietojen

siirto-ongelma koskettaa sitä läheisesti. Kiinnostava jatkotutkimuskohde, josta olisi selvää käytännön hyötyä, on varman ja aukottoman tietojensiirtorutiinin suunnittelu ja rakentaminen seuranta- ja hälytysjärjestelmineen tämän varmistamiseksi. Latausajon kaatumisesta on helppo tehdä hälytys, mutta ongelmana tietojen siirrossa on usein tiedon puuttuminen; ajosta, jota ei ajeta ei ole helppo antaa hälytystä. Koska tietojen siirtorytmit eri järjestelmistä vaihtelevat, on tällaisen keskitetyn seurantajärjestelmän rakentaminen vaativaa.

TUTKIMUKSEN LÄHTEET

Anahory S., Murray D. Data Warehousing in the real world: a practical Guide for Building Decision Support Systems. Addison-Wesley, Harlow, England, 1997.

Ault M., Oracle8, Administration and Management. Wiley and Sons, 1998.

Bobrowski S., Oracle8 Architecture. McGraw-Hill, Berkeley, 1998.

Bouzeghoub M., Fabret F., Matulovic-Broqué M., Modelling Data Warehouse Refreshment Process as a Workflow Application. Proceedings of the International Workshop on design and Management of Data Warehouses (DMDW'99), Germany, 1999.

Data Warehousing. Viitattu 6.7.2000. Saatavilla muodossa: <URL: http://www.infomanager.fi/dataware.htm >

Data Warehousing fundamentals Volume 1-3. WWW Course Guide 50102GC11, Oracle Corporation, 1996.

Date C.J.,An Introduction to Database Systems, Volume 1, Fifth Edition. Addison-Wesley, Harlow, England, 1990.

Devlin B.,Data Warehouse : From Architecture to Implementation. Addison-Wesley, Harlow, England, 1997.

DmDirect: Business Intelligence Newsletter, Challenges of efficient data cleansing By Arkady Maydanchik, September 15, 1999.

Hackathorn R.,Web Farming for the Data Warehouse. Morgan Kaufman, San Francisco, 1999.

Hovi A., Data Warehousing – Tietovarastointitekniikka. Suomen ATK-kustannus, Helsinki, 1997.

Inmon W.H., Building the Data Warehouse. Second edition , Wiley & Sons, New York, 1996.

Inmon W.H., Managing the Data Warehouse. Wiley & Sons, New York, 1997a.

Inmon W.H., J.,Zachman J.A.,Geiger J.G.,Data Stores, Data Warehousing, and the Zachman Framework : Managing Enterprise Knowledge. McGraw-Hill, New York, 1997b.

Jaakohuhta H.,Suuri tietotekniikan tietosanakirja, käsitteistö ja sanasto. Suomen atk-kustannus, Jyväskylä 1999.

Jarke M., Lenzerini M., Vassiliou Y., Vassiliadis P., Fundamentals of Data Warehousing. Springer, Berlin, 2000.

Järvinen P., Tietotekniikan termit, pc-tietosanakirja. Teknolit, Jyväskylä, 1999.

Lehtonen J., Leinonen T.,Tietovarastotekniikan perusteet, kolmas uusittu painos, Atk-instituutti, Helsinki, 1991.

Kelly S., Data Warehousing, the route to mass customisation. Wiley & Sons, New York, 1996.

Kimball R., Ralph Kimballin artikkelit dbms Magazinessa

Kimball R.,The Data Warehouse Toolkit : Practical Techniques for Building Dimensional Data Warehouses. Wiley & Sons, New York, 1996.

Mattison R.,Data Warehousing and Data Mining for telecommunications. Artech House, Boston, 1997.

Mattison R.,Web Warehousing and Knowledge Management. McGraw-Hill, New York, 1999.

Mattison R.,Data Warehousing, strategies, technologies and techniques. McGraw-Hill, New York ,1996

Meyer D., Cannon C.,Building a Better Data Warehouse. Prentice Hall PTR, London, 1998.

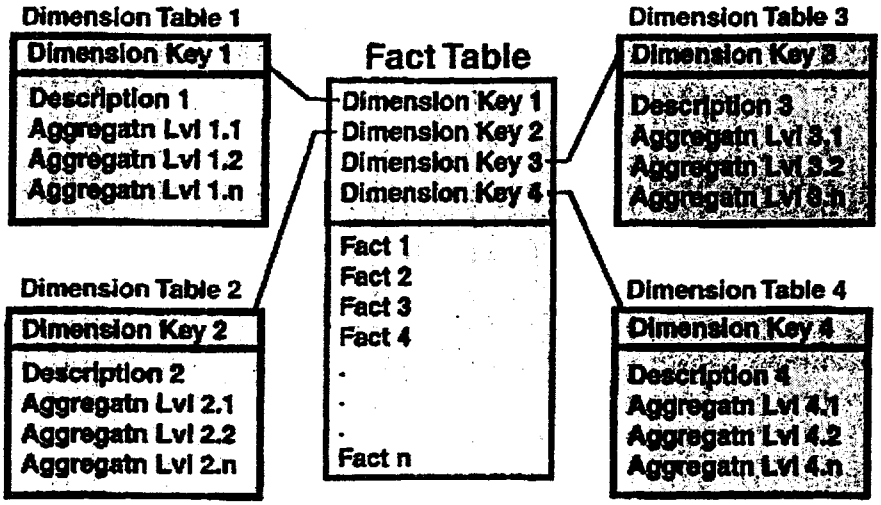
Relaatiotietokantasanasto. Viitattu 5.7.2000. Saatavilla muodossa: <URL: http://www.cs.Helsinki.FI/relaatiosanasto/>

Sainio A.,Tietovarastotekniikan perusteet, viides uudistettu painos. Yliopistopaino, Helsinki, 1998.

Web Farming for the Data Warehouse. Viitattu 26.9.1999. Saatavilla muodossa: <URL:http://www.webfarming.com/>

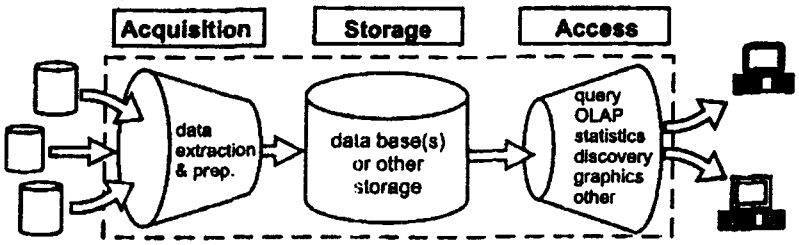
LIITTEET

LIITE 1. Star shema -tietorakennemalli (ks.kuvio 1)



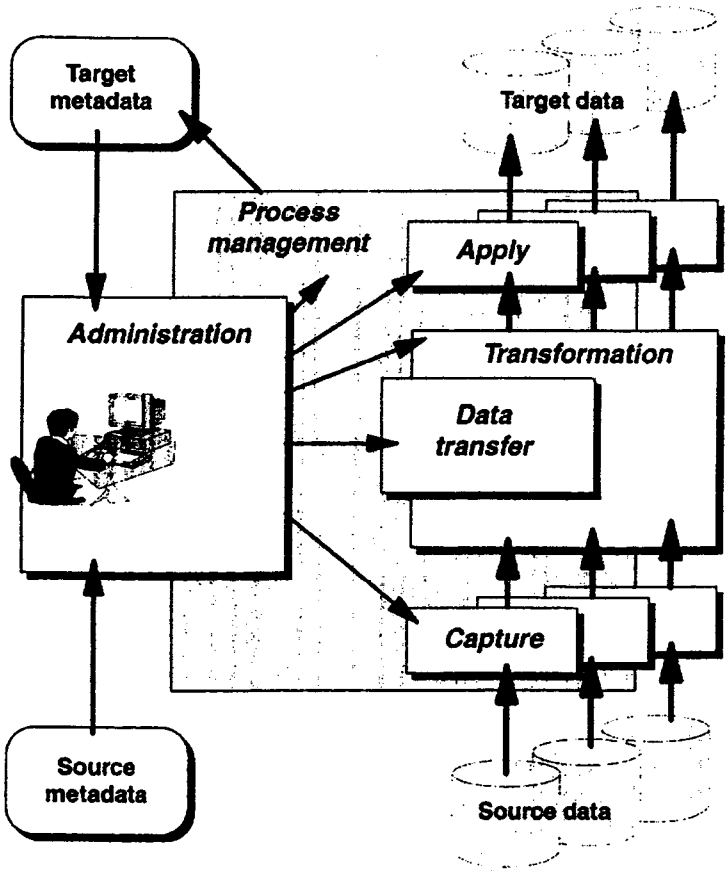
Lähde: Inmon, 1997a, s.218

LIITE 2. Tietovaraston kolme komponenttia (ks. kuvio 5)



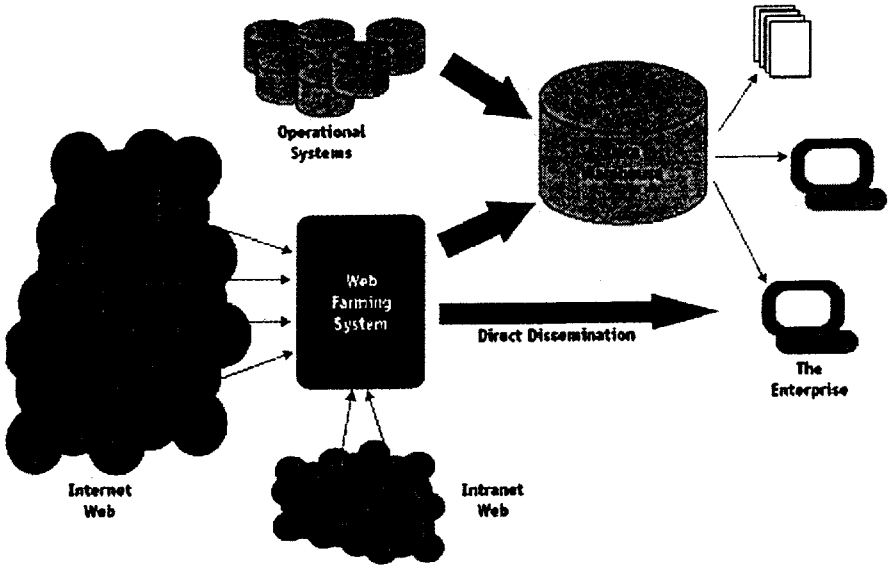
Lähde: Mattison, 1999, s.153

LIITE 3. Tietovaraston toiminnalliset komponentit (ks. kuvio 6)



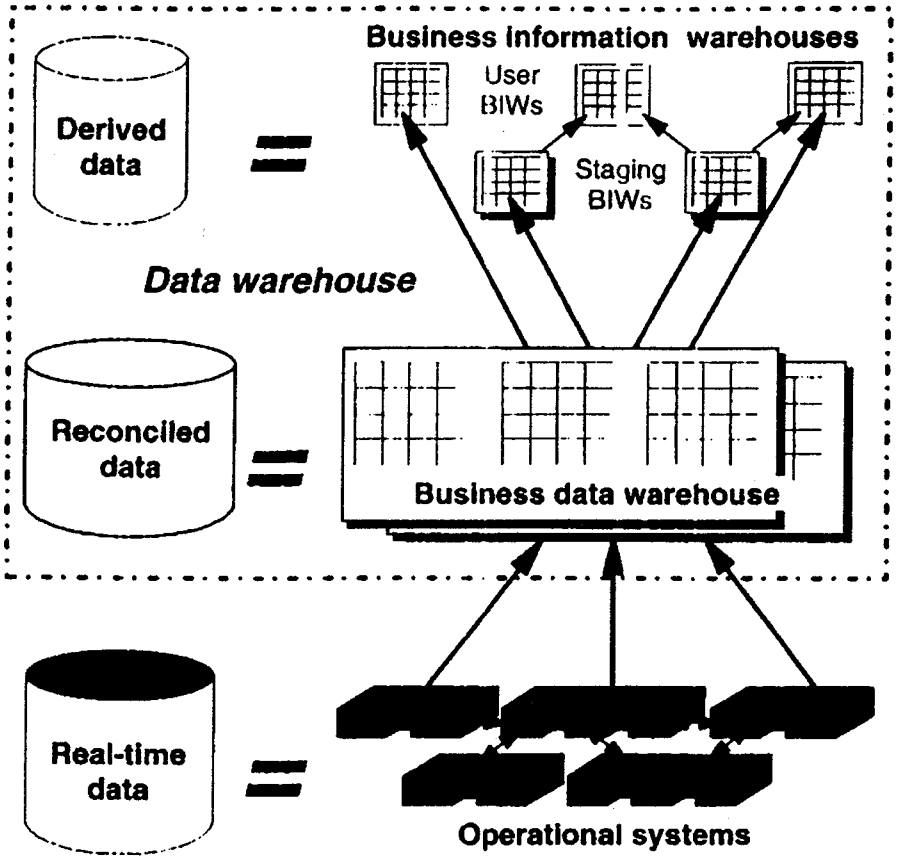
Lähde: Devlin,1997 ,s.122

LIITE 4. Tiedon kerääminen internetistä (ks. kuvio 7)



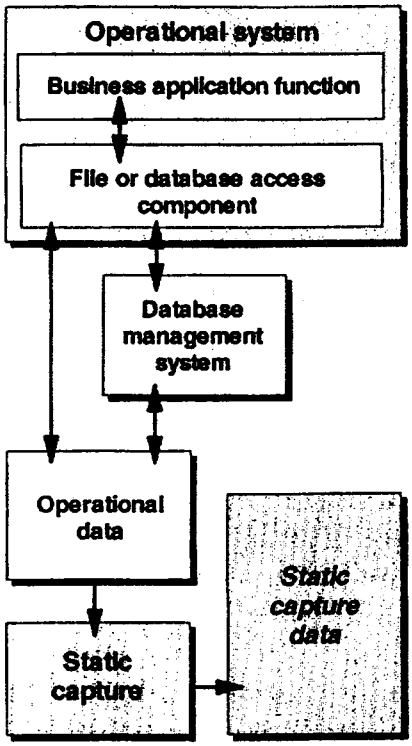
Lähde: Saatavilla muodossa: <URL:http://www.webfarming.com/> Viitattu 26.9.1999.

LIITE 5. Kolmitasoinen arkkitehtuuri ja tietovarasto (ks. kuvio 8)



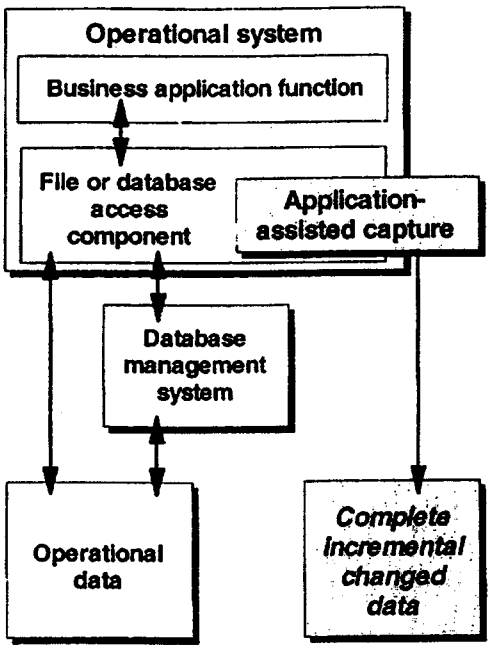
Lähde: Devlin, 1997, s. 127

LIITE 6. Muuttumaton poiminta (ks. kuvio 9)



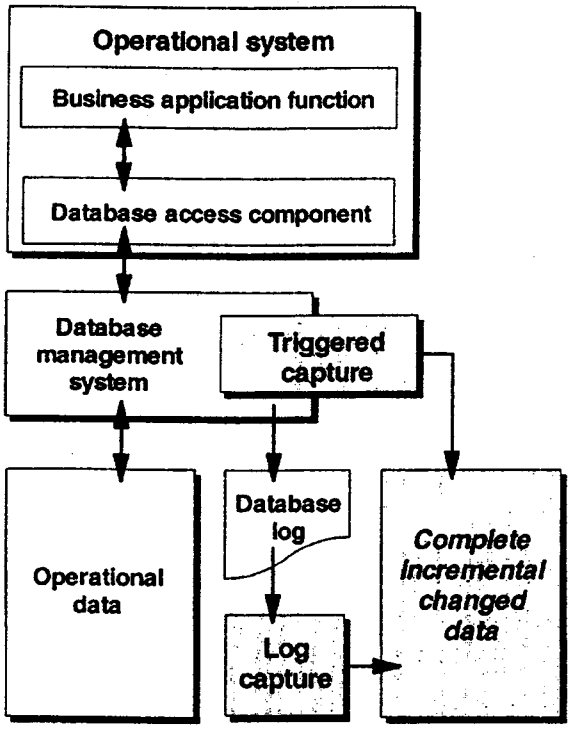
Lähde: Devlin, 1997, s. 183

LIITE 7. Sovellusavusteinen tiedon poiminta (ks. kuvio 10)



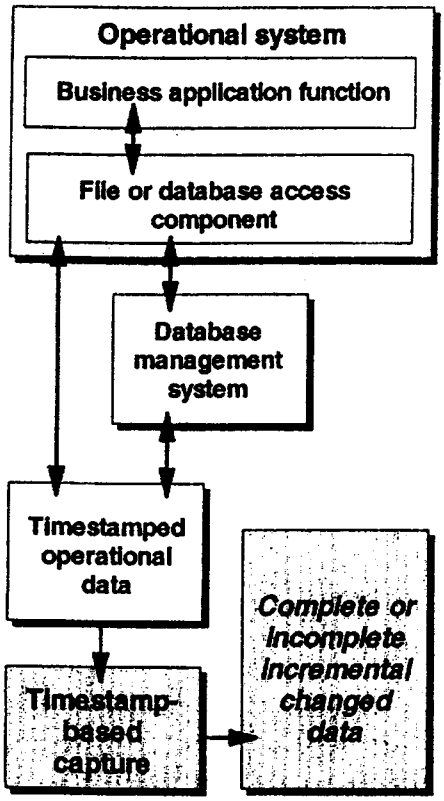
Lähde: Devlin, 1997, s. 184

LIITE 8. Liipaistu ja lokiin perustuva poiminta (ks. kuvio 11)



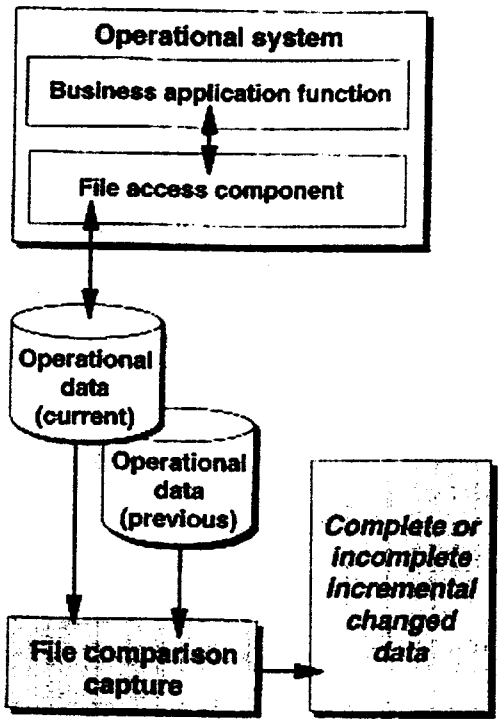
Lähde: Devlin, 1997, s. 186

LIITE 9. Aikaleimaan perustuva poiminta (ks. kuvio 12)



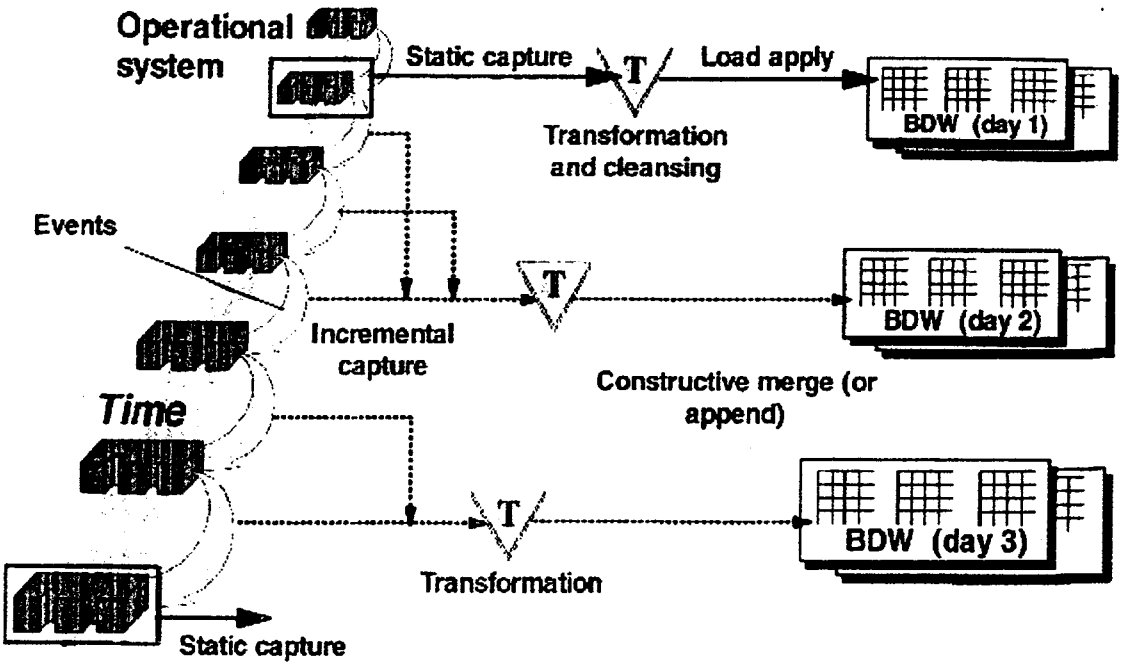
Lähde: Devlin, 1997, s. 188

LIITE 10. Tiedoston vertailuun perustuva poiminta (ks. kuvio 13)



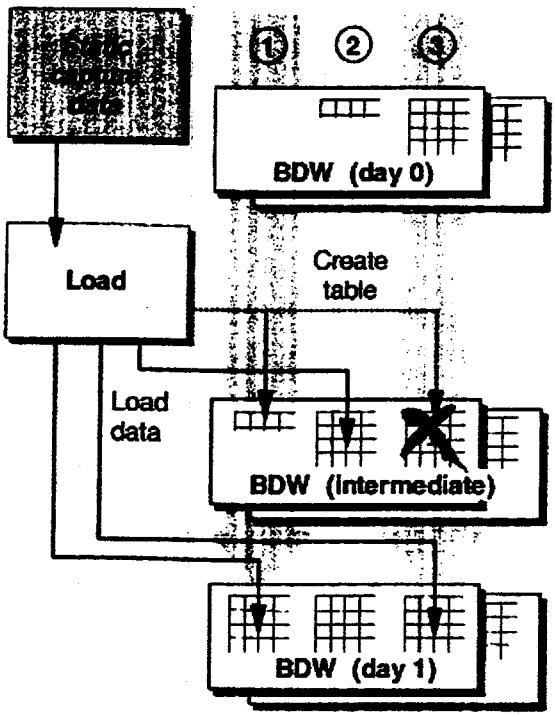
Lähde: Devlin, 1997, s. 190

LIITE 11. Tietojen poiminta, puhdistus, siirto ja lataus tietovarastoon (ks. kuvio 14)



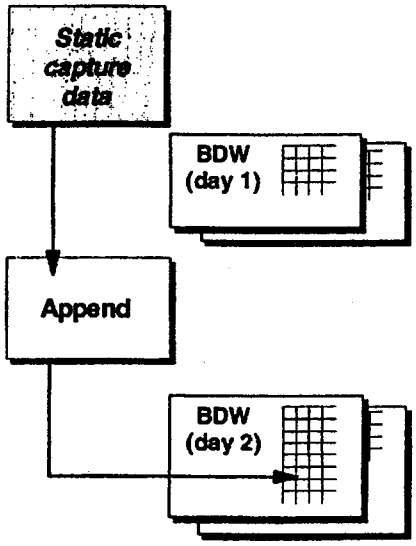
Lähde: Devlin, 1997, s. 218

LIITE 12. Alkulatauksen tallennustapa: lataus (ks. kuvio 15)



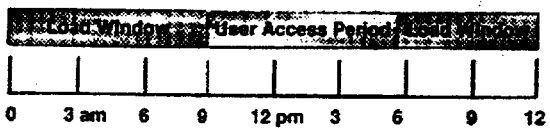
Lähde: Devlin, 1997, s. 198

LIITE 13. Alkulatauksen tallennustapa: lisäys (ks. kuvio 16)



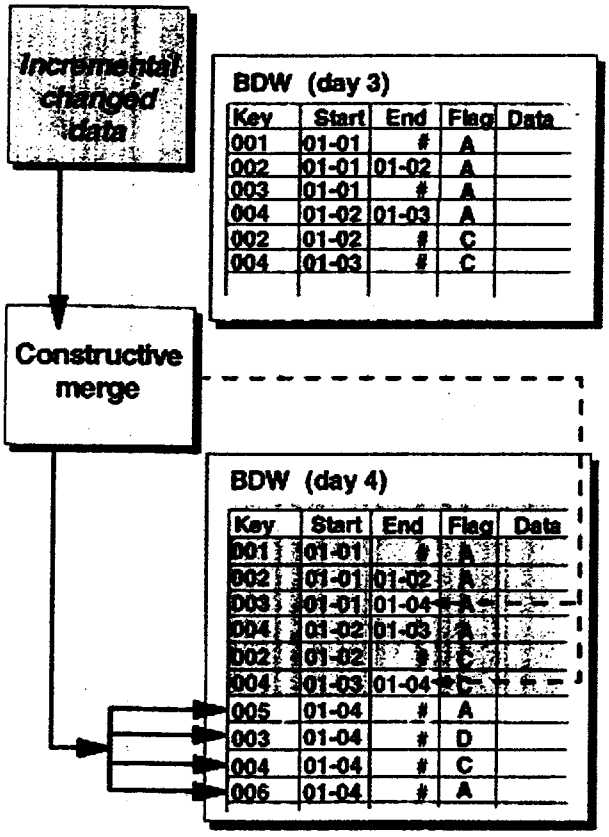
Lähde: Devlin, 1997, s. 199

LIITE 14. Latausikkuna (ks. kuvio 17)



Lähde: Data Warehousing fundamentals Volume 1-3, 1996, s. 9-14

LIITE 15. Jaksottaisen latauksen tallennustapa: rakentava liittäminen (ks. kuvio 18)



Lähde: Devlin, 1997, s. 202